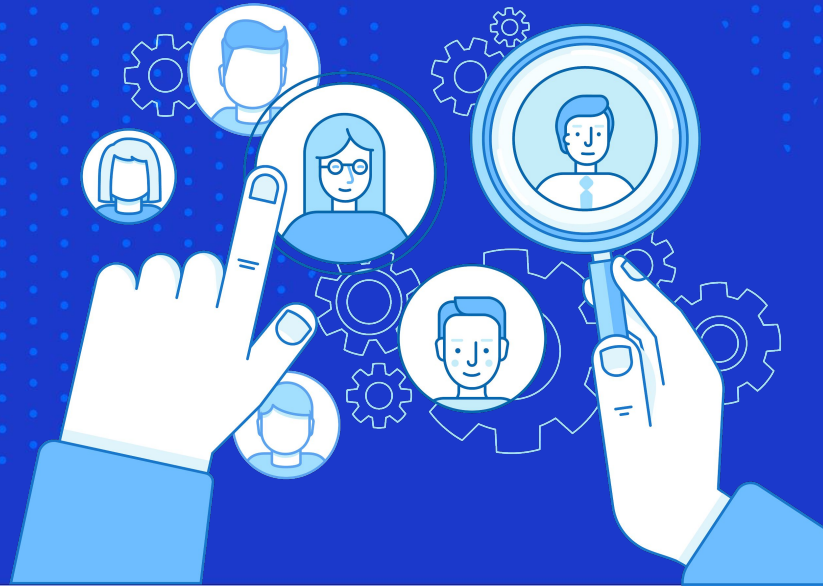


Fundamentals Module

Users and Installation & Introduction to Versioning



Topics

- Users, Groups and Permissions
- The 'sudo' command
- Versioning concept: "LTS"
- Version Control Systems
- VS Code
- Git

Users Groups and Permissions

What are Linux Users and Group Permissions?



Linux was designed to allow more than one user to have access to the system at the same time.

1

Read: a readable permission allows the contents of the file to be viewed. A read permission on a directory allows you to list the contents of a directory.

2

Write: a write permission on a file allows you to modify the contents of that file. For a directory, the write permission allows you to edit the contents of a directory (e.g. add/delete files).

3

Execute: for a file, the executable permission allows you to run the file and execute a program or script. For a directory, the execute permission allows you to change to a different directory and make it your current working directory.

Linux Primary Groups

A primary group is the default group that a user account belongs to. Every user on Linux belongs to a primary group.

The output contains each user's primary group ID.

In the example output, the primary group ID for example_user is 1001:



```
cat  
/etc/passwd
```

```
postfix:x:106:113::/var/spool/postfix:/usr/sbin/nologin  
example_user:x:1000:1001:,,,:/home/example_user:/bin/bash
```



You can also find a user's primary group information by using the `id` command.

Replace `example_user` with one of your own system's users.

```
id example_user
```

Your output will resemble this example, which displays the primary group as `example_group`.

```
uid=1000(example_user) gid=1001(example_group) groups=1001(example_group),27(sudo)
```



Linux

Secondary Groups

Once a user has been created with their primary group, they can be added to secondary groups.

A Linux system's groups are stored in the `/etc/group` file.

To find the group(s) a user belongs to, run the following command:

```
groups example_user
```



What is the difference between Primary and Secondary groups in Linux?

- A primary group is the group a user belongs to by default. Every user must belong to a primary group and a user can only belong to one primary group.
- Any new directories or files created by a user are automatically associated with a user's primary group.
- A secondary group is a group that a user is added to after their user account is created. A user can belong to zero or more secondary groups.



Creating and Deleting User Accounts

- To create a new standard user, use the `useradd` command.
- The syntax is as follows:

```
useradd <name>
```

- You need to set a password for the new user by using the `passwd` command. Note, you need root privileges to change a user password.
- The syntax is as follows:

```
passwd <username>
```



sudo command



sudo command

The Linux sudo command stands for Super User Do.

1

It is equal to the option "**run as administrator**" in Windows.

2

If we prefix the command along with other commands, it would execute that command with high privileges.

3

The sudo option allows us to have more than one administrator.

- **The `sudo apt-get update`** will figure out what the latest version of each package and dependency is, but will not actually download or install any of those updates.
- **The `sudo apt-get upgrade`** command downloads and installs the updates for each outdated package and dependency on your system. You have a chance to review the changes and confirm that you want to perform the upgrades.
- **The `sudo apt autoremove`** removes the unnecessary packages.



Versioning concept: LTS

- LTS is an abbreviation for “Long Term Support”.
- A new Ubuntu Desktop and Ubuntu Server is released every six months.
- You get free security updates for at least 9 months on the desktop and server.
- A new LTS version is released every two years.
- There is no extra fee for the LTS version; upgrades to new versions of Ubuntu are and always will be free of charge.



Version Control Systems

Version Control Systems

Version control, known as source control, is the practice of tracking and managing changes to software code.

tracks every
change

solves
conflicts

works on any
platform

facilitate a
smooth and
continuous
flow

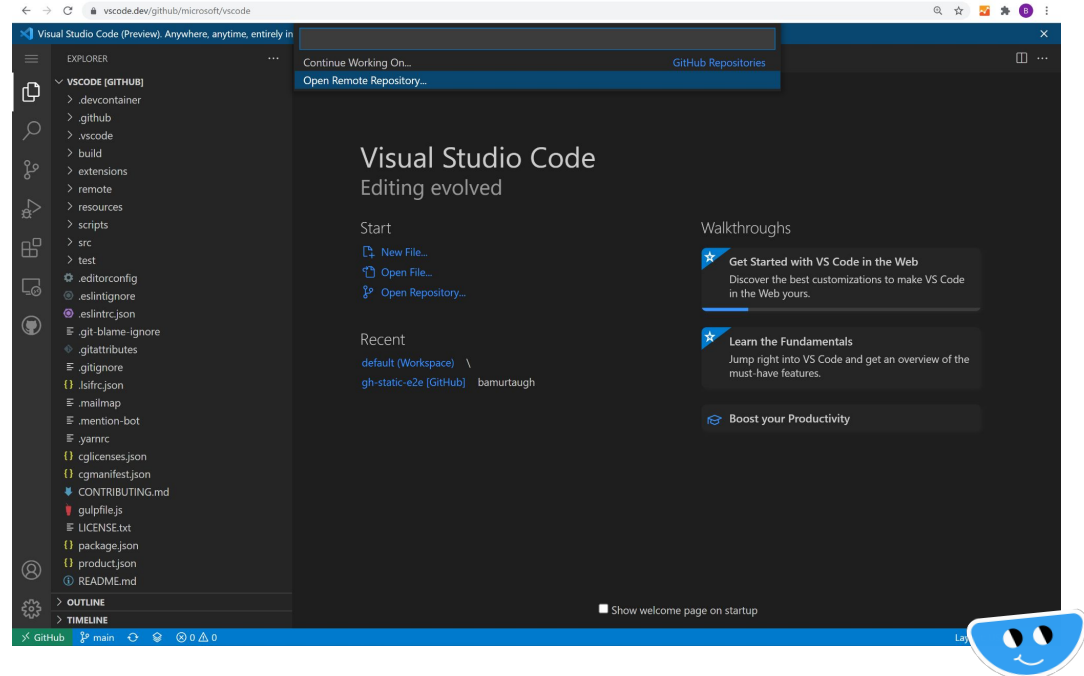
works in small
projects as
well

preserves
efficiency and
agility as the
team grows



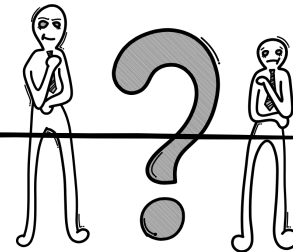
Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS.

Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.



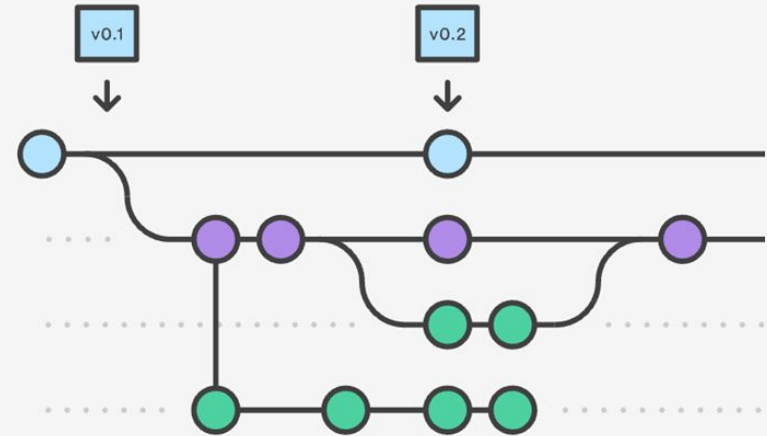
<https://code.visualstudio.com>

Your task is to research about different version control systems, decide which one is the best and explain why?



What is Git?

- Source control, or version control, is a way of tracking your files progress over time.
- It is usually saved in a series of snapshots and branches, which you can move back and forth between.



Source control allows you to:

- Distribute your file changes over time
- Prevent against data loss/damage by creating backup snapshots
- Manage complex project structures



- Git is a source control software, similar to many others out there.
- It follows all the same rules and concepts that any source control follows.



Why use Git?

- The most popular source control software in the world
- Lots of documentation and support
- Lots of integration with other applications (SourceTree, Heroku, GitHub)



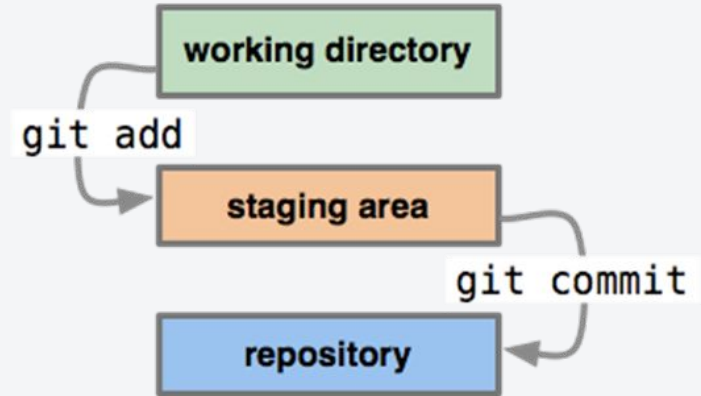
- Before Git tracks a change, it goes through a long chain of operations and tasks.
- Many of these tasks are user controlled, and are required for changes to be tracked correctly.



- Repositories, usually called 'repos', store the full history and source control of a project.
- They can either be hosted locally, or on a shared server, such as GitHub.
- Most repositories are stored on GitHub, while core contributors make copies of the repository on their machine and update the repository using the push/pull system.
- Any repository stored somewhere other than locally is called a 'remote repository'.

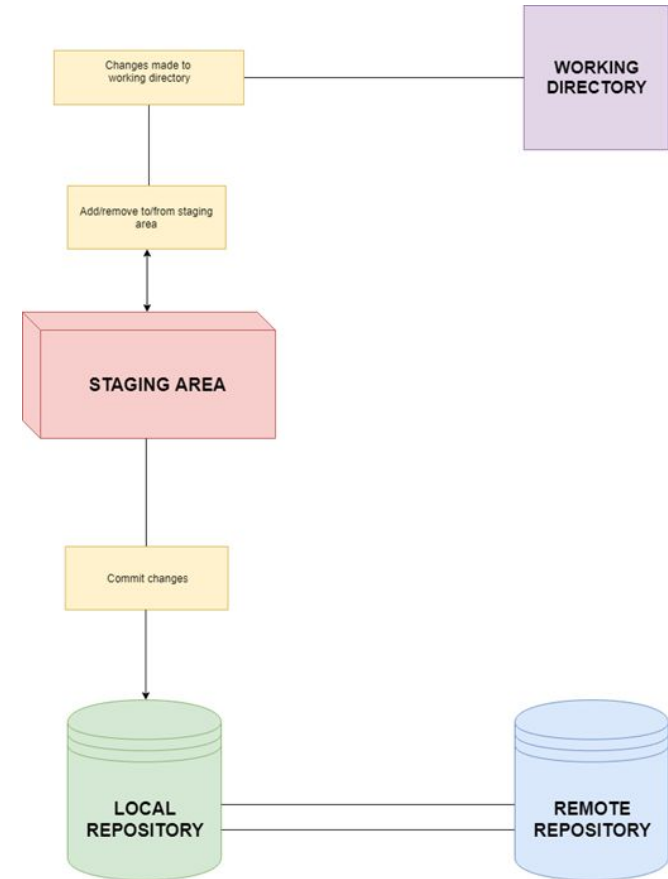



- Repositories are timelines of the entire project, including all directories, or 'working directories' are projects at their current state in time.
- Any local directory interacting with a repository is technically a repository itself, however, it is better to call these directories 'local repositories', as they are instances of a remote repository.




Repos vs Directories

- This diagram shows a little bit about how the basic Git workflow process works
- The staging area is the bundle of all the modifications to the project that are going to be committed.
- A 'commit' is similar to taking a snapshot of the current state of the project, then storing it on a timeline.





**At the end of this lesson,
you should be able to
understand the
following topics:**

- Linux Users, Groups and Permissions
 - The 'sudo' command
 - Versioning concept: "LTS"
 - Version Control Systems
 - VS Code
 - Git
- 

Objective:

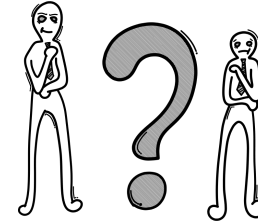
Get familiar with Git commands.

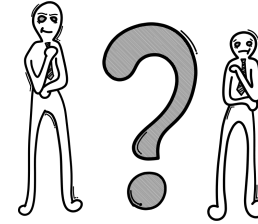
Your task:

All of the following tasks should be performed using terminal

- Create a folder named NewApp
- Create only one file within the NewApp folder
- Initialize the NewApp folder as a git repository
- Add the file you created to the staging area
- Create 10 new files within NewApp folder
- Add all the files to the staging area using one command
- Show the status of your staging area
- Untrack two files from the staging area

Please, don't close your terminal.



**Objective:**

Get familiar with Git commands.

Your task:

All of the following tasks should be performed using terminal

- Remove two other files from the staging area
- Commit all your changes using "Added new files" message
- Edit the content of three files, three times and commit the changes separately
- Show all the commits you did along with their IDs
- Chose one of the files and bring back the first version of that file.

Please, don't close your terminal.

Objective:

Get familiar with Git Branches.

Your task:

All of the following tasks should be performed using terminal

- Create a new folder
- Initialize it as a git repository
- Create two new folders within your new folder
- In each of those two folders create 4 new files
- Show the current git status
- Add all the files and folders to the staging area
- Create your first commit with the message "Initial commit"
- Create a new branch and switch to it
- Get back to the master branch
- Create a another new branch
- List all the branches that you currently have
- Delete one of the branches
- Choose one of the branches, switch to that branch, modify files and then commit the new changes
- Show the list of all the commits that you made
- Choose two branches and merge them
- Show the branches you merged.



Thank you!

*Have a great day
ahead!*

