MERN Stack Training / Web Full Stack / Javascript Full Stack

Duration: 140 hours

MongoDB: NoSQL Database to maintain the data

Express.js: Backend services

React.js: Frontend

Node.js: Runtime environment

Contents:

* Applications
* Fundamentals of Computer Programming
* Algorithms & Pseudocode
* Linux
* Git & Git Hub
* HTML
* CSS
* Javascript & Typescript
* React.js
* Node.js
* Express.js (REST Api’s)
* MongoDB
* DevOps

Applications: They are computer programs which can perform tasks for the users, there are 2 types of applications

1. Standalone
2. Distributed

Standalone applications: These are applications which you can used only after running in the machine

Two types

* Desktop based: This can be installed on your desktop
* Mobile based: This can be installed on your mobiles

ex: Word document, Mobile based applications

Distributed applications: These are the applications which you can access over the internet

ex: Banking services, Twitter, Facebook, Gmail, Food Ordering Systems, Shopping cart applications

How do we access distributed applications

We can access them using Browser or Client applications

Browser: It is an application which helps to enter URL and use the distributed application

Client applications: These are the applications that can internally use the URL to access distributed application

ex: Mobile applications, ATM machine programs, Swiping Machines programs

URL: Uninform Resource Locator, it is a name given to the applications to easily access it, in real time the applications use IP address & Port numbers, but users can’t remember it hence, URL will be given

Browser:

Browser is an application which helps to access any websites through URL, there are many browsers

1. Google Chrome
2. Edge
3. Mozilla
4. Opera
5. Older browsers like
   1. World wide web browser
   2. Mosaic
   3. Internet explorer

Distributed Applications:

These are the applications that can be accessed over the internet, these applications will be running in some remote machines, there are mainly 2 types of Distributed applications

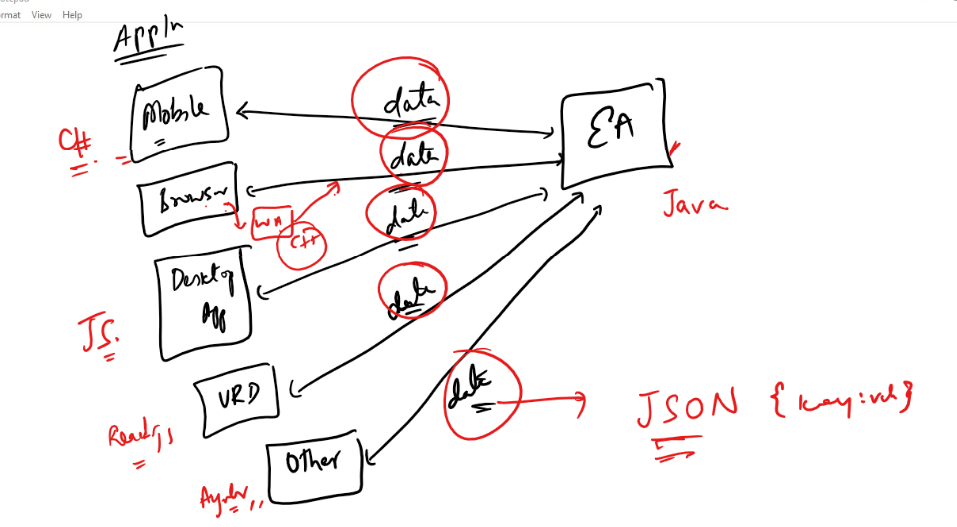
1. Web applications
2. Enterprise wide applications

Web application:

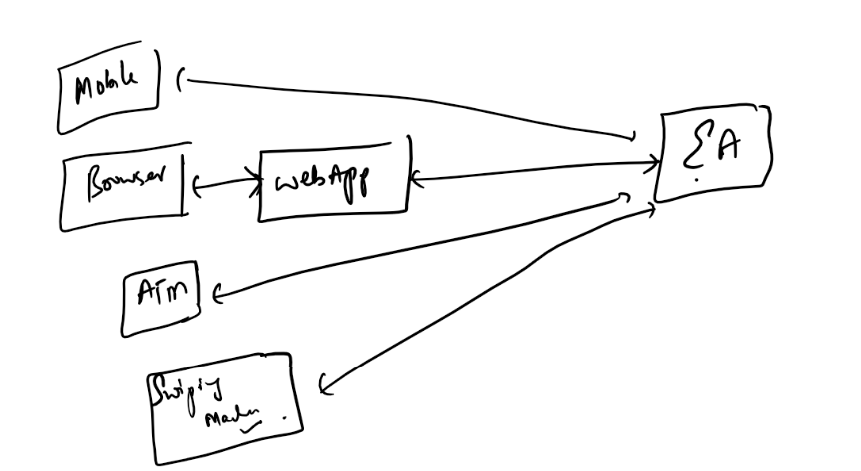
A web application takes request from the browser & returns the response in the format the browser understands i.e., HTML, CSS & Javascript

Enterprise application:

These are business related applications which can serve various types of applications written in different languages.



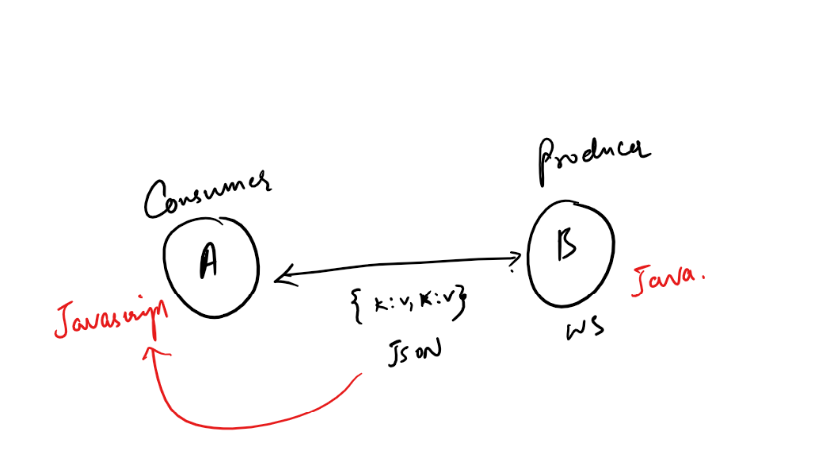
The enterprise application return data in a format which every application can convert to their respective types, suppose Java wants to share data to C#, then Java wouldn’t share data in Java instead it shares in one common format JSON that will be converted to C# by C# applications, same can be converted to Javascript by Javascript applications, same can be converted to Python by Python applications.



How does these enterprise applications share/exchange data to other applications

Enterprise applications will have webservices (ReST based Webservices) to share the data to any applications.

REST WebServices: These are online services or API’s which are made available on the internet so that any applications can use their URLs to consume the data, usually data will be in JSON format which is converted to the format the consuming application is written in



What are the common formats the webservices can use to share the data

1. JSON (More widely used): Easy to understand & maintain hence its used widely almost 99.9% of the time
2. XML
3. TEXT
4. HTML

Web applications: These applications will have web pages, there are 2 types of web pages

1. Static web pages: These are the pages which are having same contents for all the users, it doesn’t change but its common for all the users

ex: Wikipedia

1. Dynamic web pages: These are the pages whose contents are different for different users, these pages change their contents at runtime

ex: Gmail pages, Facebook pages and so on

Technologies/Languages used to create Static & Dynamic web pages

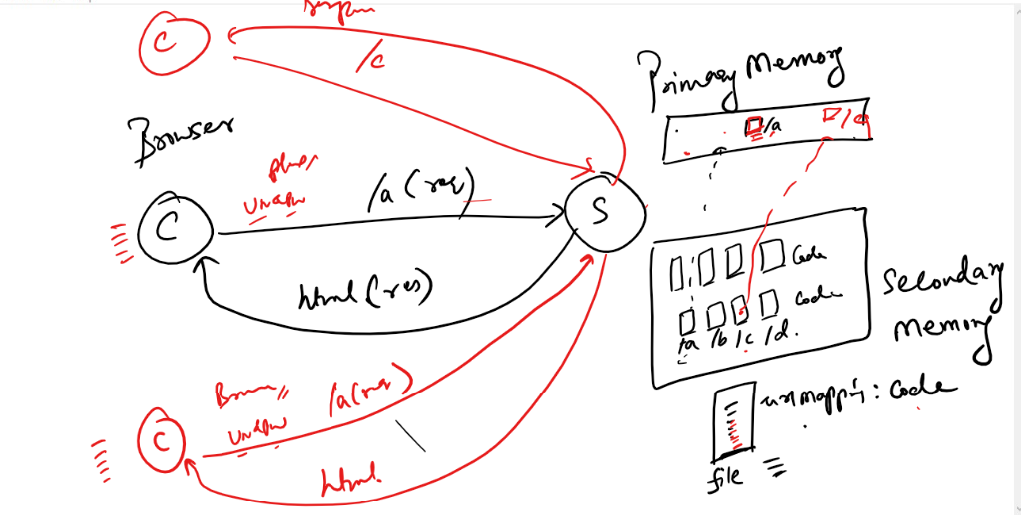
Static pages are created with below languages

1. HTML
2. CSS
3. Javascript

Dynamic pages are created with various technologies

1. C#: ASP .NET
2. Java: Servlets & JSP
3. Python: Django
4. Javascript: React.js, Angular.js, Ember.js

How the dynamic pages are created



Here when client sends the request to the server, it sends the URL to the server, the server is going first check in the primary memory whether any instance/entity/object can handle the request if the entity is not present then server checks in some configuration file to know which is the entity to handle the URL, it loads that entity to the primary memory & it executes the code to generate response, here response will be HTML because client is browser, however it can generate different HTML response for different clients for same URL.

ex: If you login to gmail then you see different page and others see different page.

What is the language browser understands

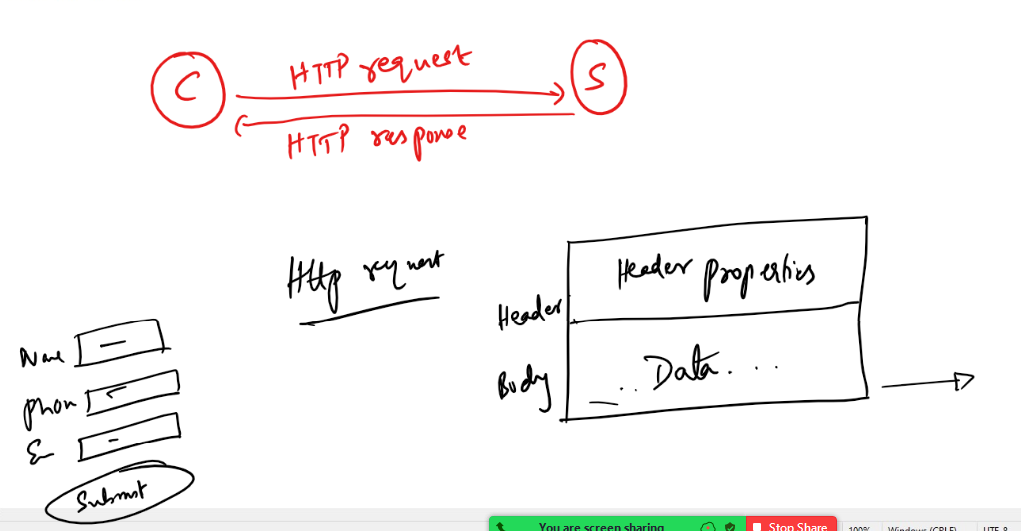
1. HTML
2. CSS
3. Javascript

What is the language client & server uses to communicate

HTTP: Stands for Hyper Text Transfer Protocol, it is the language client & server uses to communicate, they communicate using request & response, since they use HTTP they are also called as HTTP request & HTTP response.

Architecture of HTTP request & HTTP response

HTTP Request



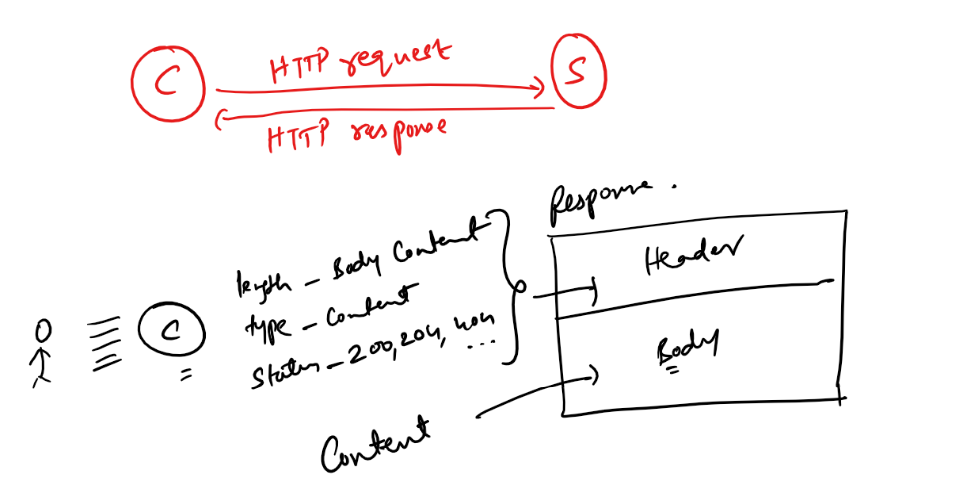
When client sends the request an HTTP request will be sent, the HTTP request will be divided into two sections

1. Header
2. Body

Header: It will have header properties like URL of the server resource, content type like text, json, xml and so on

Body: it will have content/data

HTTP Response



The Response will have 2 sections header & body

Header of the response will have header properties of response like content type, content length, time take to produce response, status codes like (200, 404, 405) and so on

Note: status code plays an important role for the client programs, because client programs can understand whether the request is succeeded or failed.

If status codes are in 2xx series then its success

If status codes are in 4xx series then its failed

HTTP status codes

These are some standard codes for client & servers used while communicating so that they can make decisions based on the status code

200: Ok

201: Created

404: Resource not found

401: Unauthorized access

400: Bad request

Application design patterns

Design Pattern:

It is a solution to the commonly occurring problems in application development, there are many design patterns in the IT industry which suits for certain kind of requirements, some of the most widely used design patterns are:

1. Singleton pattern
2. MVC pattern

Singleton pattern: It is a design pattern which creates one & only instance to serve some tasks, even if you try to create multiple instances it gives you single instance

ex: Task manager of an OS doesn’t create multiple copies of it, instead same copy of task manager is used

ex: Database Connections can be singleton so that multiple instances of the application can use same database connections

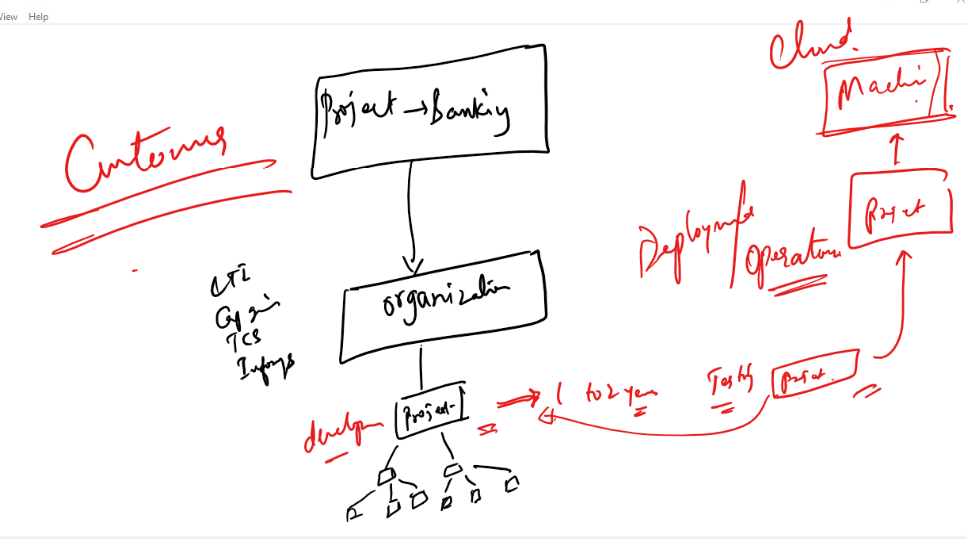
MVC pattern: MVC stands for Model View Controller, it is one of most widely used design pattern in many of the applications almost all the applications follow it, because it separates the roles of the code so that it will be easy to maintain and develop.

This makes application to easily adapt for the changes and development will be faster because you will have different teams for different roles like one team for only view layer, one team for only controller layer and another team for only model layer.

What type of application we will develop in Full Stack

A fullstack developer needs to know all the phases (developing, testing, deploying) of the application development, full stack deals with two types of applications

1. Front end applications
2. Back end applications



Backend applications:

These are the backend services which are available online to access from any kind of applications normally they are RESTful webservices, they will have database connectivity, business logics, data processing logics and so on.

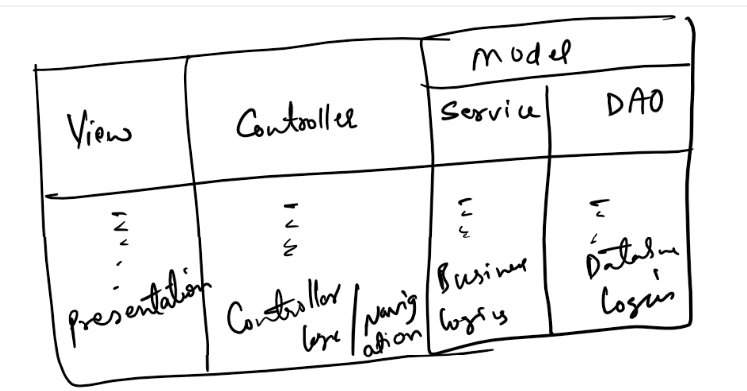
Database logics will take care of performing operations on the database, normally these operations are called as CRUD operations(Create Retrieve Update Delete), However business logics will take care of performing business specific actions.

Business logics:

These are the logics which are performed before database logics, like

1. checking for balance before debiting the amount, else respond with insufficient amount
2. Checking for ID card validity to process the user details
3. In Facebook you need to wait for 24 hours to again delete your account if in case previously you have deleted and cancelled it

When we follow MVC architecture you will keep in business logics & database logics in model layer as below:-



We have many technologies to implement the backend services like

* Java
* C#
* Python
* Javascript (node.js & express.js)

Front End applications:

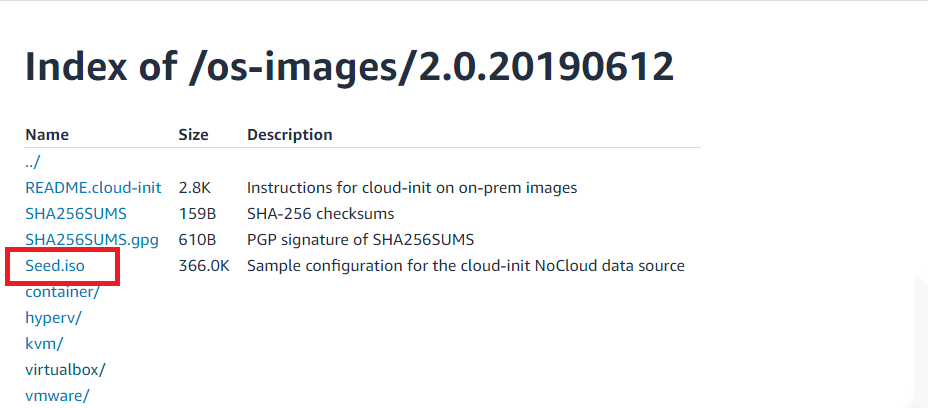
These are the applications which can be run on different devices they will have only UI’s it could be Mobile, Web Browser, Different devices like Watch, VRH, GPS devices, swiping machines, they can be developed in various technologies like

* Java for android devices
* React.js for many mobile applications
* Angular Framework
* Ember.js
* Javascript

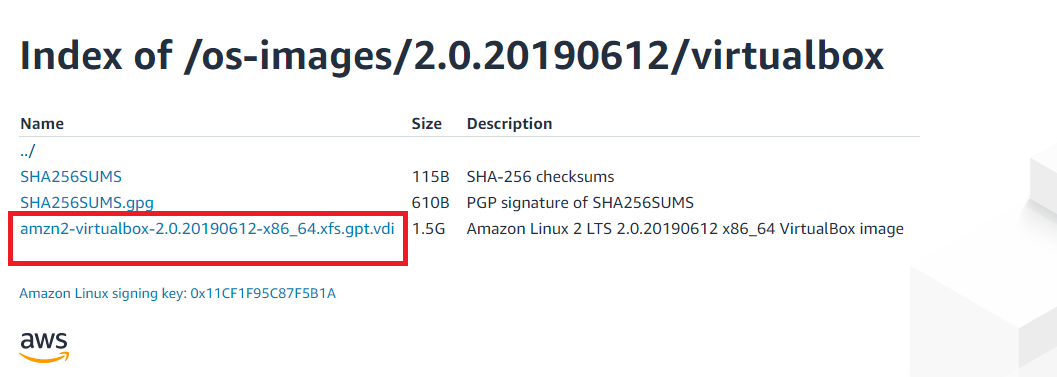
Installing Linux

* You can install it using virtual box, it is not a base machine
* You need to get two files from the below website

<https://cdn.amazonlinux.com/os-images/2.0.20190612/>



Click on the virtualbox folder



The above highlighted is an image for Virtual box that helps us to create a virtual machine to install linux OS

Lastly you need to download & install Virtual Box software

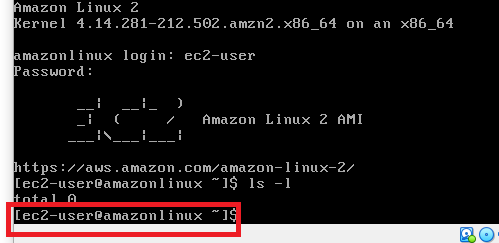


Steps to Install Linux in base machine using virtual box

1. Firstly we need to Virtual Box
2. Secondly we need to open Virtual Box & install Linux OS

Once you login to the linux machine you will see below message as EC2 Amaxon Linux 2

The highlighted part is the location of the linux command prompt, i.e.,   
[ec2-user@amazonlin]$



Linux commands plays an important role in our course

* Git
* React & Nod.js: When you use terminals to launch react & node applications
* Deployment

Evolution Programming Languages

Computer understands Machine Level Language, when it comes to language evolution we have

1. Machine Level Language: Hard to understand
2. Assembly Level Language: Works only with certain processors
3. C: Procedure Oriented Programming Language
4. C++: Object Oriented Programming Language & Platform dependent
5. Java: Object Oriented Programming Language & Platform Independent
6. Javascript: We can use it to develop both frontend and backend applications

How to create applications

Applications are created with programs, these programs will have lot of rules in the form syntax to make compilers & machines to understand, we need to follow those rules to make the application work, else we get errors

We must able to write the code properly, but for that as a beginner we must know 3 things before writing the code

1. Algorithms
2. Flowchart
3. Pseudocode

Before writing the code / program we must know how to solve the problems, hence we have algorithms, flowcharts & pseudocode

All these three are independent of the programming languages as they are not understood by any compilers or machines, they are steps given for humans to understand what is the solution for the problem & use those steps they can write the program.

Algorithms:

* It is independent from the language
* It is a step by step procedure to solve a given problem to get the desired result
* It is not understood by computers
* It gives steps in English statements

Flowcharts:

* It is independent from the language
* It is to understand the flow of the problem solution to better understand the Algorithms
* It gives us pictorial representations of the algorithms
* It is not understood by computers

Pseudocode:

* It is independent form the language
* It mimics programming languages
* It gives us the idea what all the common programming constructs you have to use to solve the problem like conditions, loops
* It is not understood by computers

Note: There is no standard rules about how we should write Algorithms/FlowCharts/Pseudocode, we can write them however we want to understand properly

Note: Once we become good at coding, we may not need these Algorithms, Flowcharts, Pseudocode

Let us write all 3 to solve a simple problem

Problem: Find the area of a circle & print the area

Algorithms

Step1: Start

Step2: Take the radius

Step3: Take the PI

Step4: Multiply radius with radius once

Step5: Multiply PI with the result you get from Step4

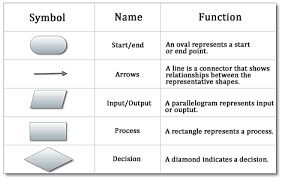
Step6: Take the result you get from Step5 and store in a variable area

Step7: Print the area

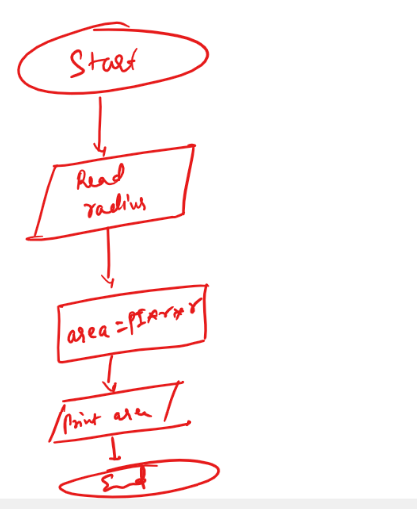
Step8: Stop

Flow charts

Flow chart has some defined symbols



Flowchart for the area of circle



Pseudocode:

It mimics the programming language, the pseudocode for the area of circle

begin

read radius, area

set radius=value

calculate area = pi \* radius \* radius

print area;

end

Simple Activity:

Find the highest number in the two numbers

i.e., which is greater number

Write Algorithms, Flowcharts & Pseudocode for it.

Duration: 10 mins

Linux:

It is one of the popular server OS mainly used to run applications, install servers, databases and so on, most of the real-world applications are running in Linux machines, because they are faster than any GUI based OS.

User can access OS’s in 2 ways

1. Through GUI
2. Through Terminal/Command Prompt

GUI based OS are good for end users, there are many GUI OS’s like Windows, Mac OS, Fedora, Ubuntu and so on.

Terminal based OS are good for business related applications that are run inside these OS’s.

Linux follows Unix OS feature, Unix is also a terminal based OS but it is not free, however Linux is free & Open source

Open Source: You can customize as per your requirement

Unix vs Linux

Unix is licensed OS, it is used in mainframes and for installing server side resources, Linux also follows same architecture of Unix, all the commands of Linux works in Unix also and vice versa.

Linux comes in 2 flavours

1. Terminal based: Meant for hosting applications, installing servers, databases and so on, ex: Amazon Linux EC2, Solaris and etc
2. GUI based: Meant for end users who needs GUI, ex: Fedora, Ubuntu

Linux commands:

These are the commands Linux based OS understand to work with Linux, you can do all kind of operations like creating files, editing files, deleting files, renaming files, installing software’s, running programs.

List of utility commands in linux

1. cal
2. date
3. pwd
4. ls
5. ls -l
6. cat
7. rm
8. mv
9. grep

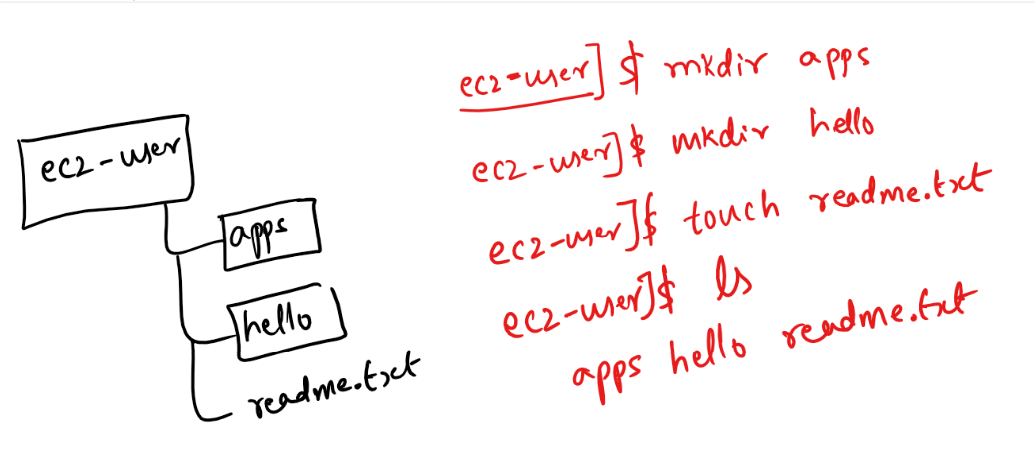
cal: Shows calendar

cal -y: Show current year calendar

cal 2015: Shows 2015 calendar

cal 05 2015: Shows May 2015 calendar

date: Shows the date



cd: It helps you to navigate from one directory to another directory

Suppose you are inside ec2-user directory and ec2-user has 2 directories

1. apps
2. hello

From [ec2-user]$ you can use

cd apps : This takes to [ec2-user/apps]$  
(or)  
cd hello: This takes to [ec2-user/hello]$

mkdir: It helps you to create directory

Suppose you want to navigate from one directory to another backward then you can use cd ..

cd ../../: This goes 2 steps backward

cd abc/xyz: This goes to xyz folder present inside abc folder which is present inside current folder

How to edit files

vi: Command used to edit files

Insert characters: You need to enter insert key to enter any characters

Exit from edit mode: You need to press Esc key & :wq

Note: You will see :wq at the bottom of the terminal

cat: It is a command to view the content of a file

rm: to remove files & folders

rm -r: to remove a non-empty folder

Ex: rm -r abc

rm -r \*: This deletes all the files & folders

Grep: Global Regular Expression Print:

This is for searching the pattern in the files

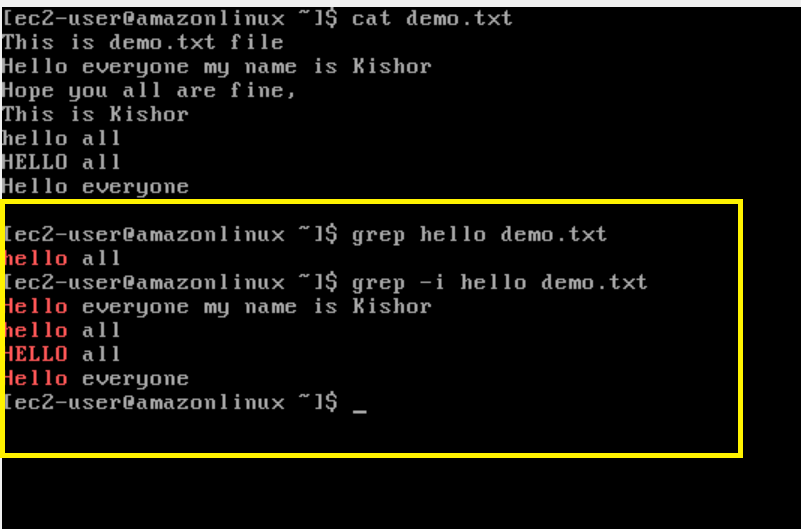
grep pattern filename: This command lists all the matching pattern in the file

grep pattern filename filename: This lists all the matching pattern in the multiple files

1. Create 2 files demo.txt & test.txt
2. Edit both the files with few lines of data and also have your names in both the files and also repeat your names a few times in both the files

Ex:

Hello everyone  
This is Alex, I’m learning Linux  
I understand this is server OS  
I’m Alex, I’m understanding Linux



Here -i ignores the case while searching.

Git:

It is a centralized version controlling system, it keeps track of the project state with different version numbers these version numbers are 40 characters hexadecimal values ranging from 0 to 9 and a to f, with this 40 characters code git can recognize all the changes like what files and folders are modified, at what time its modified, who modified and so on.

Git uses following things to control the versioning of our work

1. Repository: A Git folder that keeps files & folders along with that a version controlling folder called .git , this repository is like a backup that will be available in everyone’s machine which are called as local repository, there are two types of repositories
   1. Remote repository that is maintained GIT hub
   2. Local repository that is maintained in every users computer who wants to push their work to the remote repository

Note: Every repository is a mirror of other’s repository

1. Branch: A branch is moveable pointer that tells us we are in which version of the work, by default Git gives one branch it could be main/master

Note: Most of the times you don’t work in main/master branch instead you will create your own branch i.e., feature branch.

Creating Repository

There are 2 repositories

1. Remote repository: Shared to everyone
2. Local repository: For a particular user which will be linked to remote repository

There are two ways you can create a repository

1. Create in Git hub(remote repository) and clone in your local machine(local repository)
2. Create in local machine(local repository) and then create in Git hub(remote repository) and manually link the local repository with remote repository

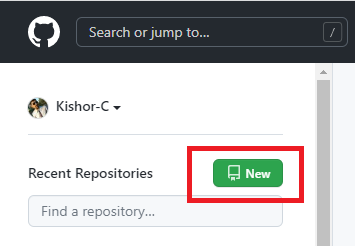
Note: Repository is created only once to get all the histories of the work, it is not created everytime, it will be created either only when you don’t have that repository.

Creating the Repository in Remote and Cloning it in the Local machine

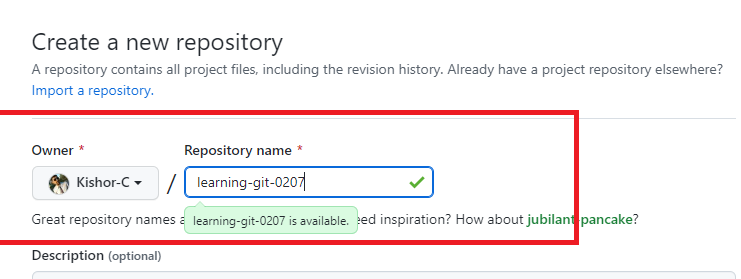
1. Create the repository with a name in the GIT hub
2. Use git clone command in a directory (ensure its not under any previous repositories) it must be a directory that is not under git versioning

Note: To know you are cloning in a non git repository just type git status and it must output you are not inside a git repository, if git status shows some output like files tracked, files untracked then you are inside git repository so you must not create another repository here.

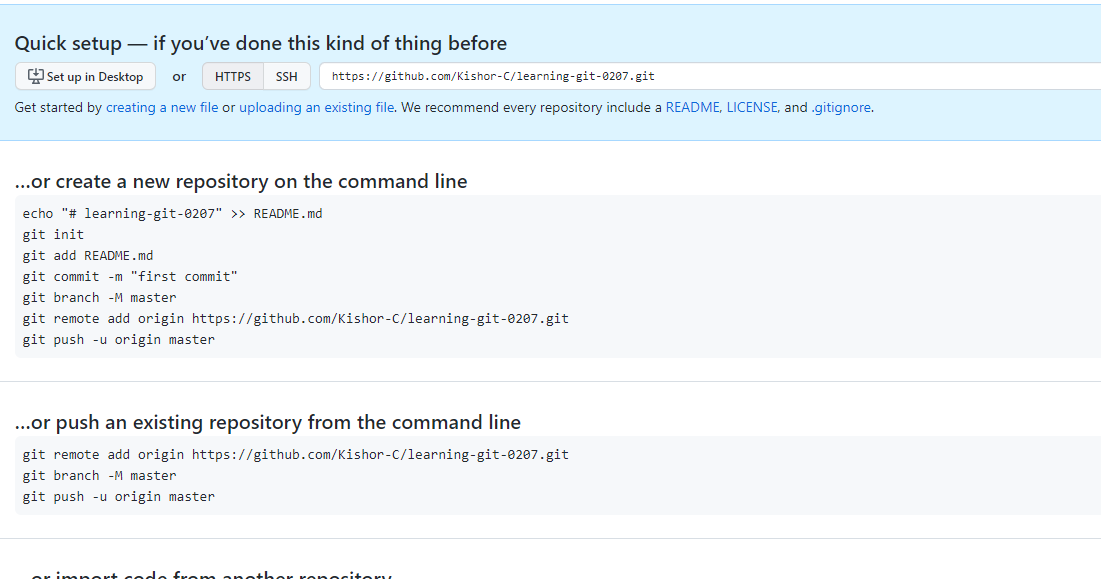
Click on new



Create a new repository by entering the name & scroll down to create

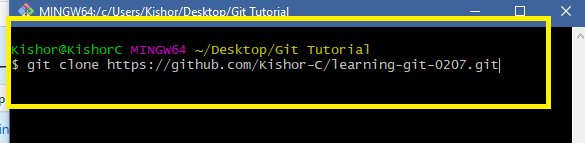


Scroll down to see create repository button & click on it, you will see the repository page as below

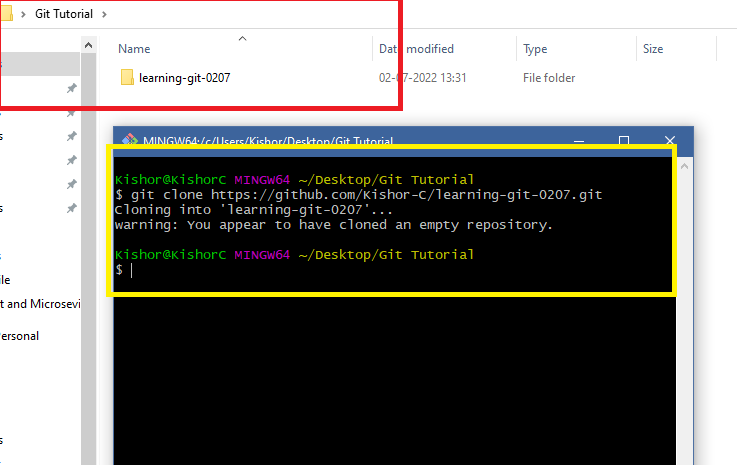


Next we need to clone this remote repository to the local machine to create local repository

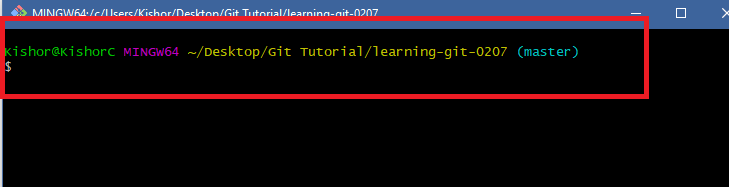
Open Git bash in a non-git repository and use ‘git clone <<url>>”



When you hit enter



Notice that you see a folder called learning-git-0207 that is empty, navigate to it, you may see main/master both are ok



Note: There’s not history in this repository because it is a new repository

Let us create our first version, it must be in the main/master branch

Do some changes and update the remote repository

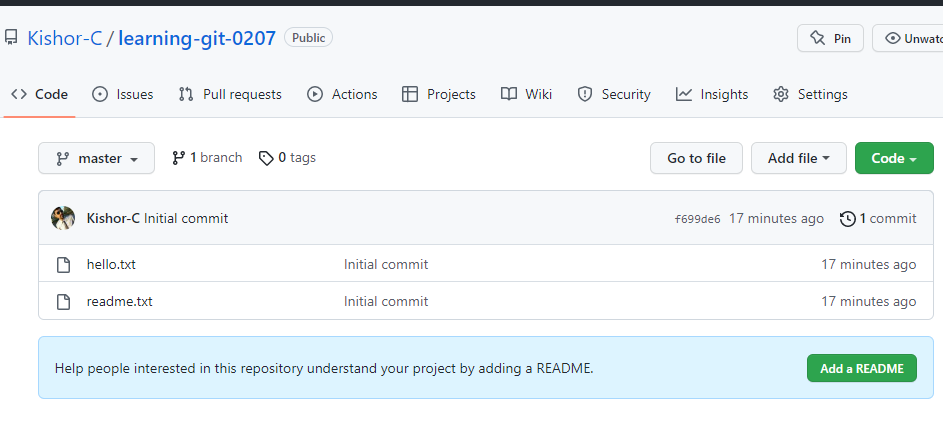
>> touch readme.txt hello.txt

>> git add .

>> git commit -m ‘initial commit’

>> git push -u origin master(or)main

You must see 2 files in the remote repository also



Edit readme.txt file and write your name in it and follow the same steps(add, commit, push) to push those changes in the remote repository

Currently you are working in main/master branch which is not advisable because it is a branch that is treated as a stable branch which will be stable in everybody’s machine.

master/main branch:

* It is treated as a original copy that needs to be same in everyone’s machine
* It is used by testers, developers, operations team to work on the project

How to list branch

>> git branch

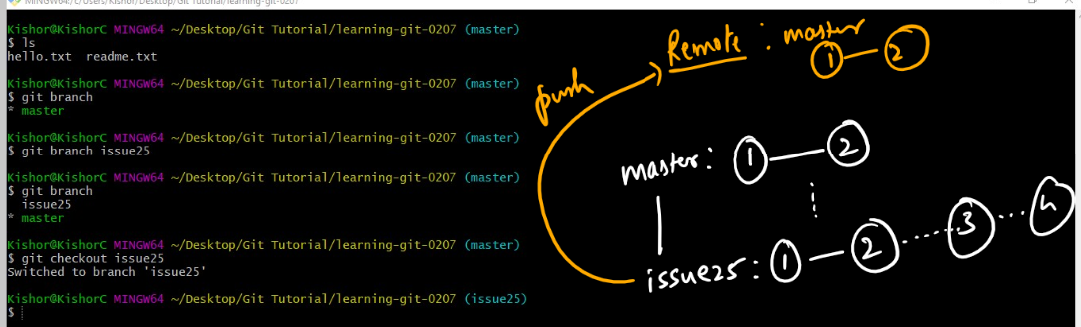
How to create a feature branch

>> git branch branch\_name

How to checkout to feature branch

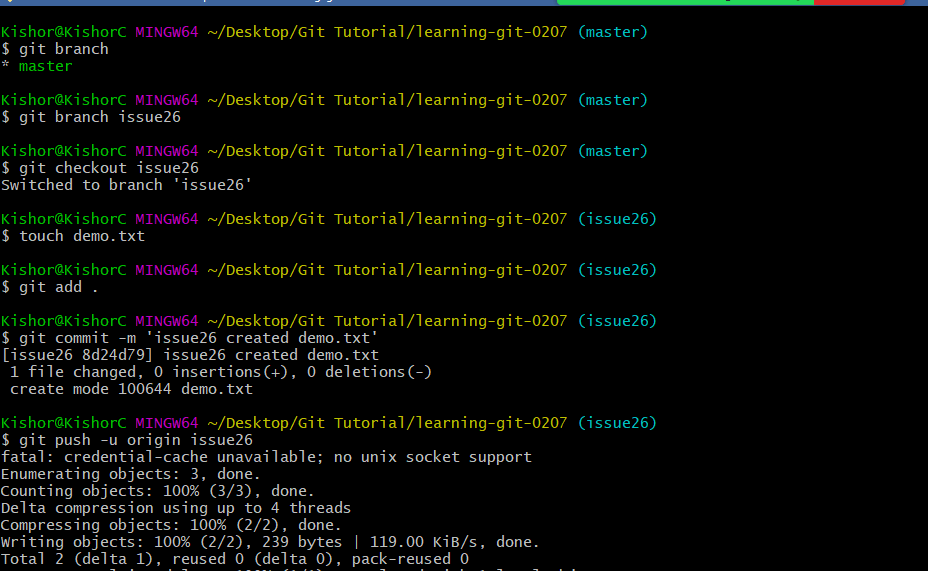
>> git checkout branch\_name

Once you checkout to feature branch you are completely independent from the master branch



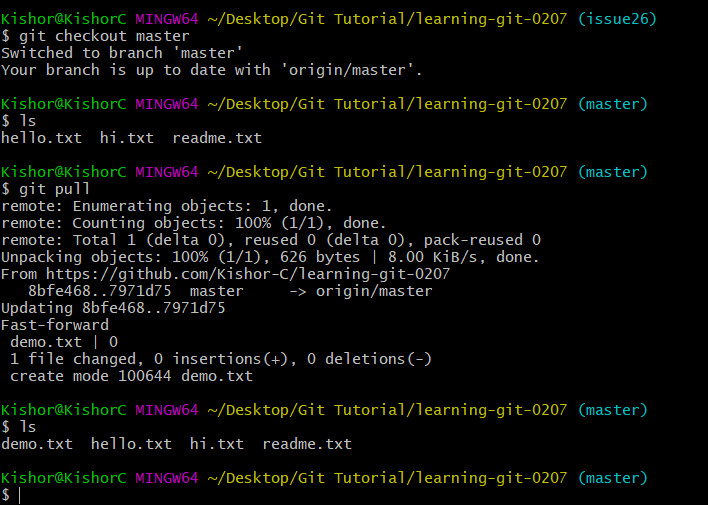
Things done with feature branch

1. git branch issue25 (creates a new branch issue25)
2. git checkout issue25 (switches to the branch issue25)
3. Make some changes in issue25 and create a commit
4. git push -u origin issue25 (pushes the issue25 changes to the remote)
5. In Remote you can merge the pull request & safely delete issue25 in both local & remote machines
6. git pull (pulls the remote master changes to the local repository & updates the local repository automatically)



Now in Git Hub merge the pull request

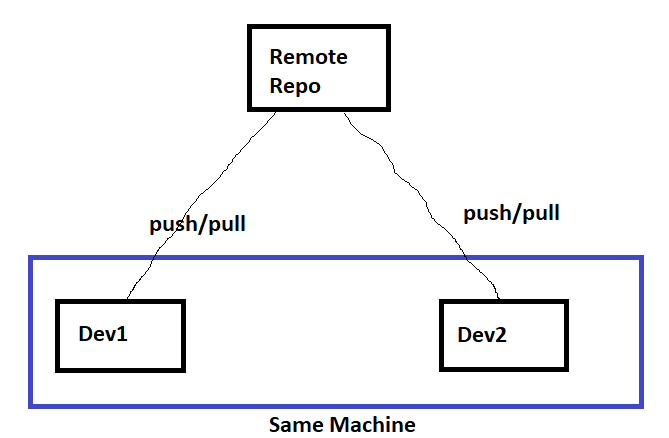
After merging, you can pull the remote changes in the local master as below



You were in issue26 and switched to master, then through ls you identified only 3 files, but after git pull you got 4 files as remote master had 4 files, now the local master & remote master both are up to date.

Working with the GIT when multiple users have conflicts with the history

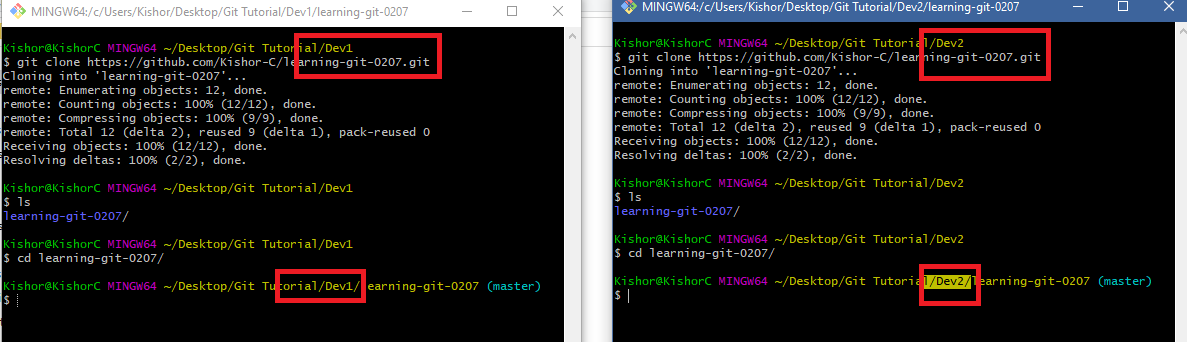
We can create 2 folders and in each folder the same remote repository can be cloned so that each folder can act as independent local repository like independent user.



Create 2 directories with the name dev1 & dev2



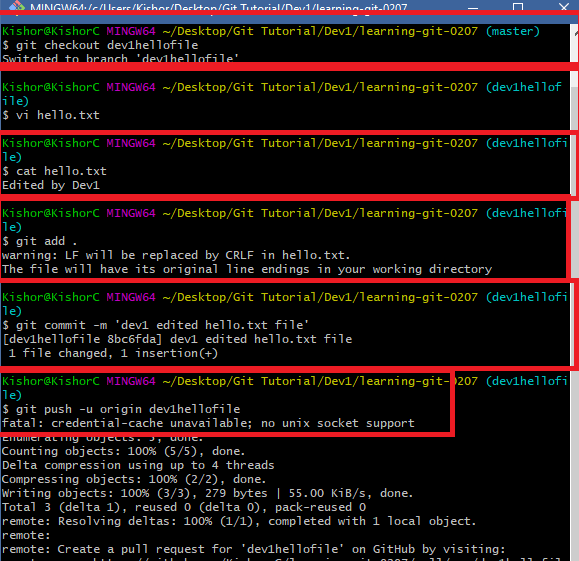
In each directory clone the same repository and navigate to each repositories



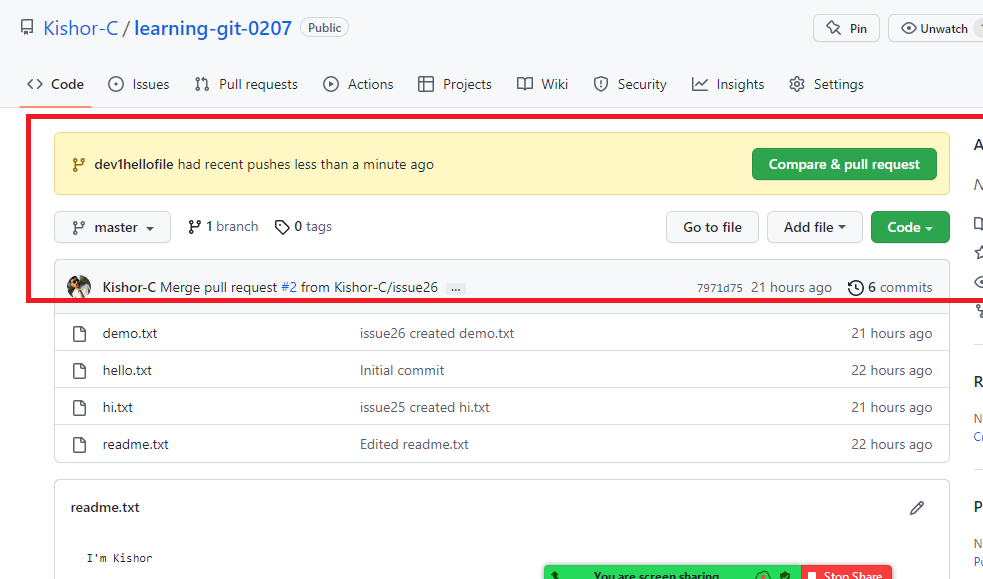
As we know that we must not work in the master branch to make any changes we need to create a feature branch and push the same to the remote

In Dev1 & Dev2 the versions are same, but now we can make changes to the same file in each directory

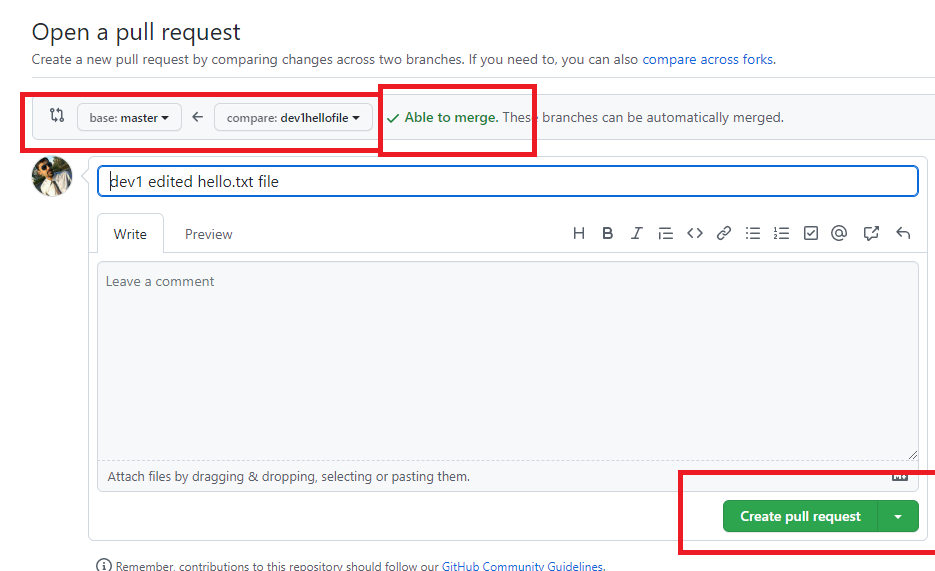
Create a new branch in Dev1 and edit hello.txt



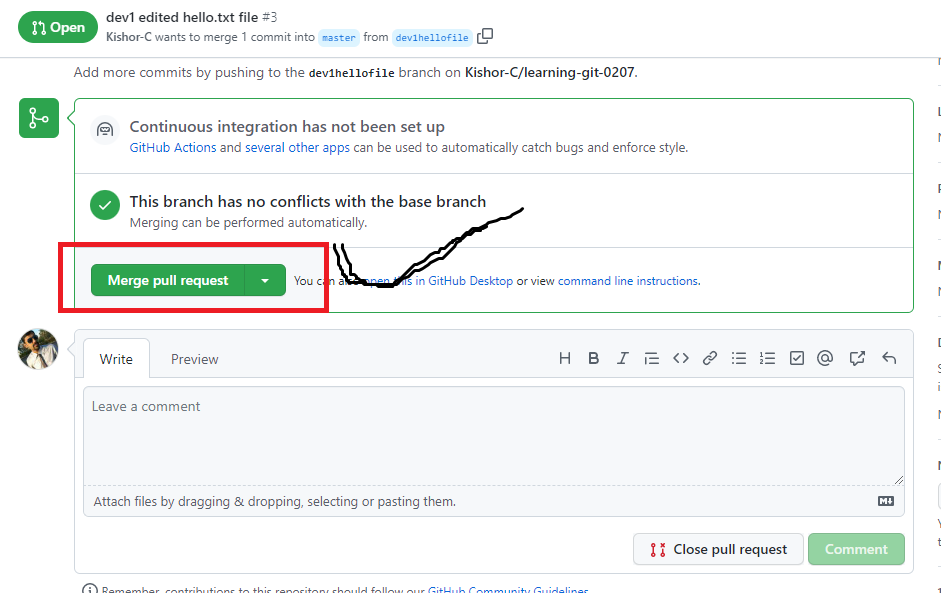
We must see the pull request in the remote server i.e., Git Hub



You can click on Compare & pull request



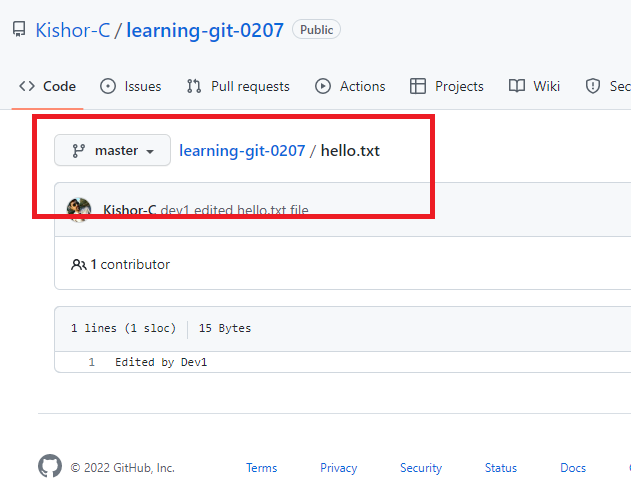
You can decide whether you can merge or simple close the pull request



Since there’s no conflict you can merge the pull request, if in case there was a conflict you must close the pull request and let user first pull the updates from the remote & then again push

Once you merge successfully you must see the content of hello.txt in the Git hub

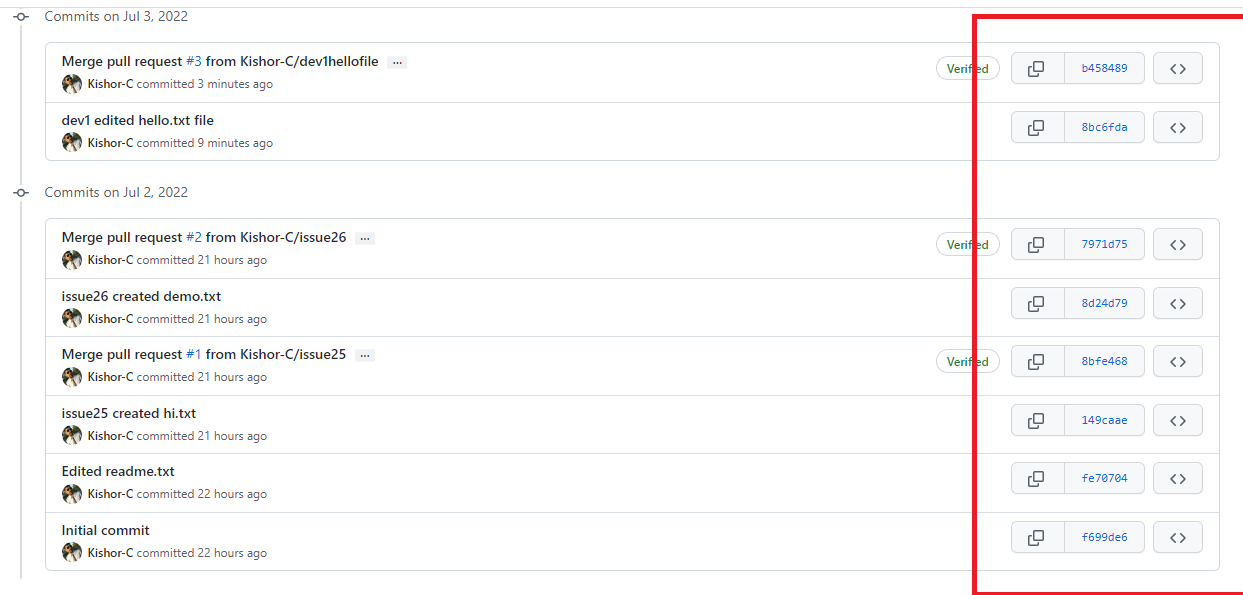
Note: After merge you can delete the feature branch in the remote server



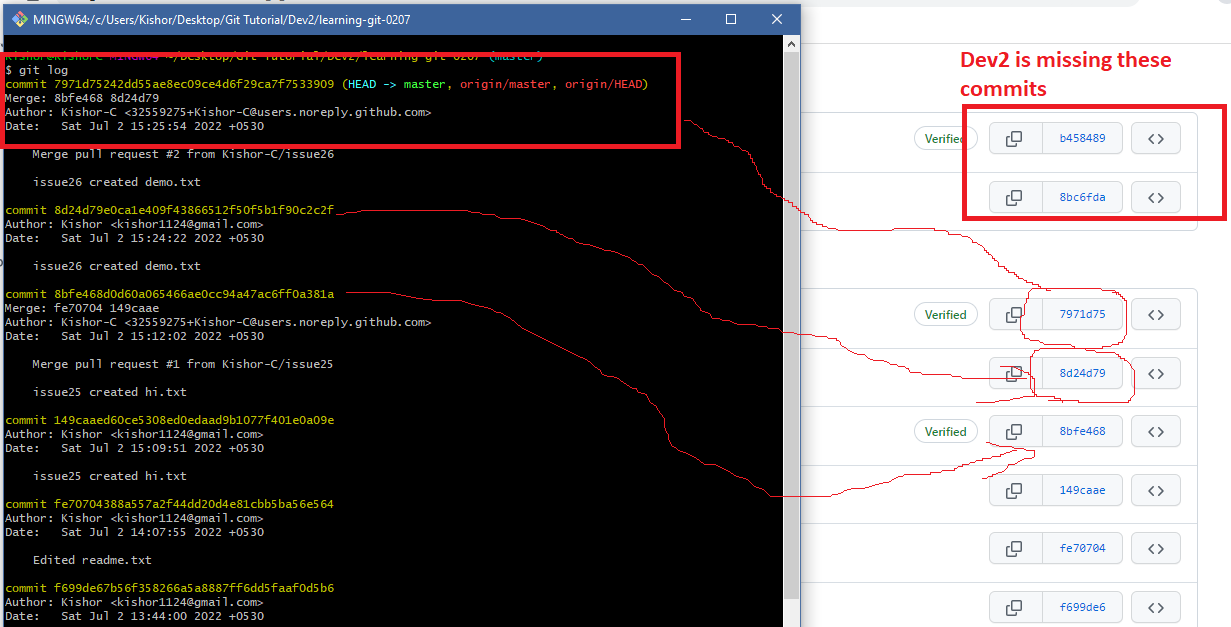
Currently Dev1 didn’t see any conflict as his branch had all the versions that were available in the remote hence his branch was able to merge.

But dev2 may face conflict because his branch is not upto date

Lets see the logs of both Dev2 & Remote master

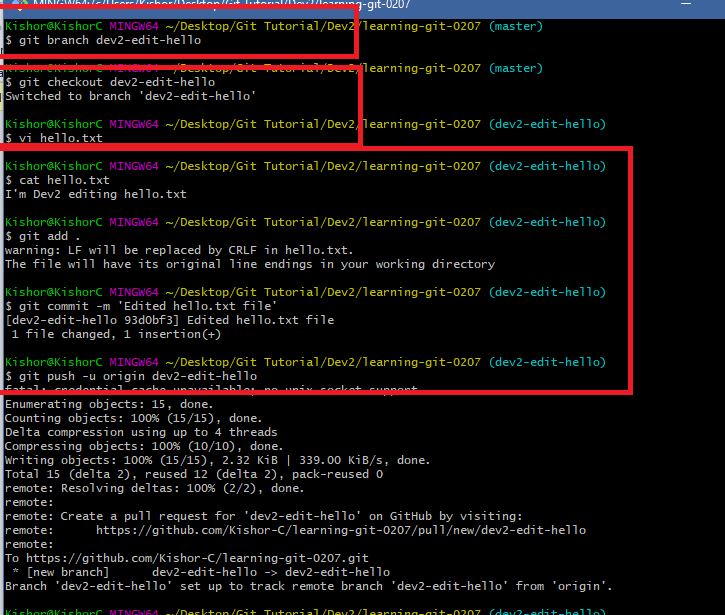


Dev2 is not having 8bc6fda…. & b458489….

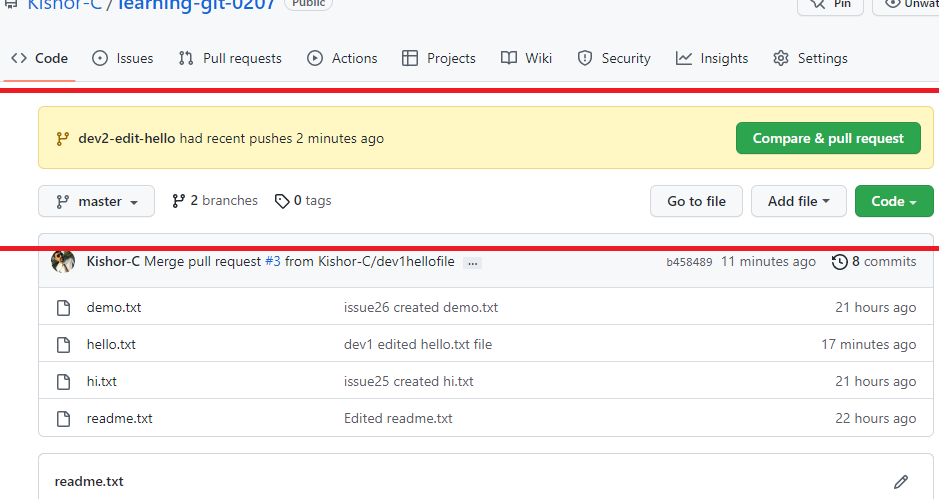


If Dev2 doesn’t pull those updates & tries to push his changes to the remote then merge conflict occurs, we will see how it occurs.

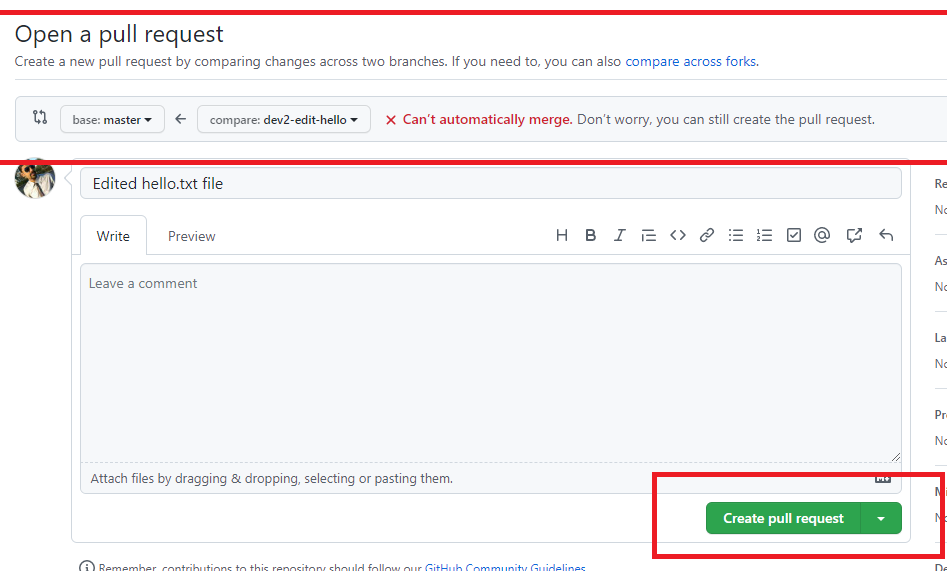
Firstly create a new branch in Dev2 & edit hello.txt file and push the feature branch to remote server



In the remote server we must see the pull request from the feature branch but it can’t be merged because there is a conflict that is what even Git also says

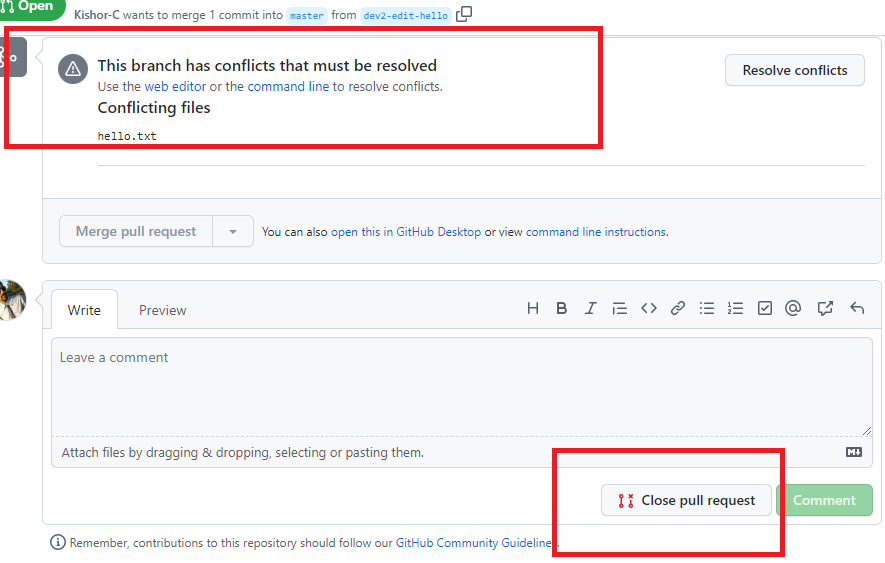


Once you click on Compare & pull request we will see merge conflict error



You will see can’t automatically merge, however you need to click on Create pull request & close the pull request so that the user who pushed this feature branch must pull the remote updates to his local repository and then he must push the feature branch.

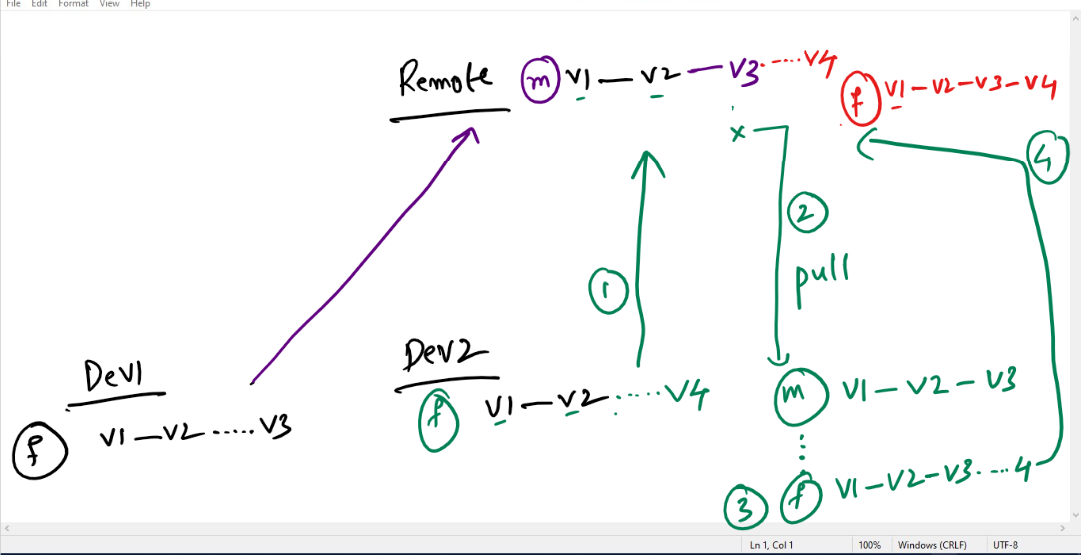
Click on Create pull request



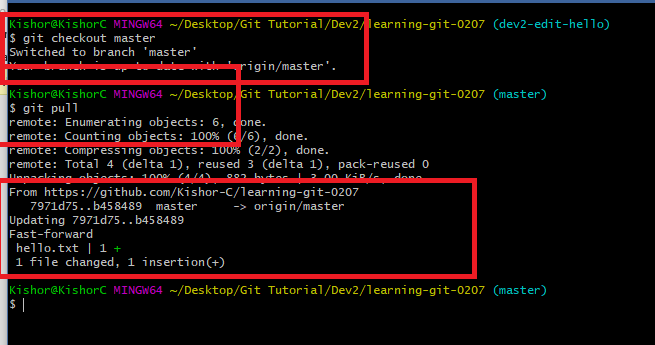
Click on close pull request

You can see branch is not merged and you must able to delete the feature branch.

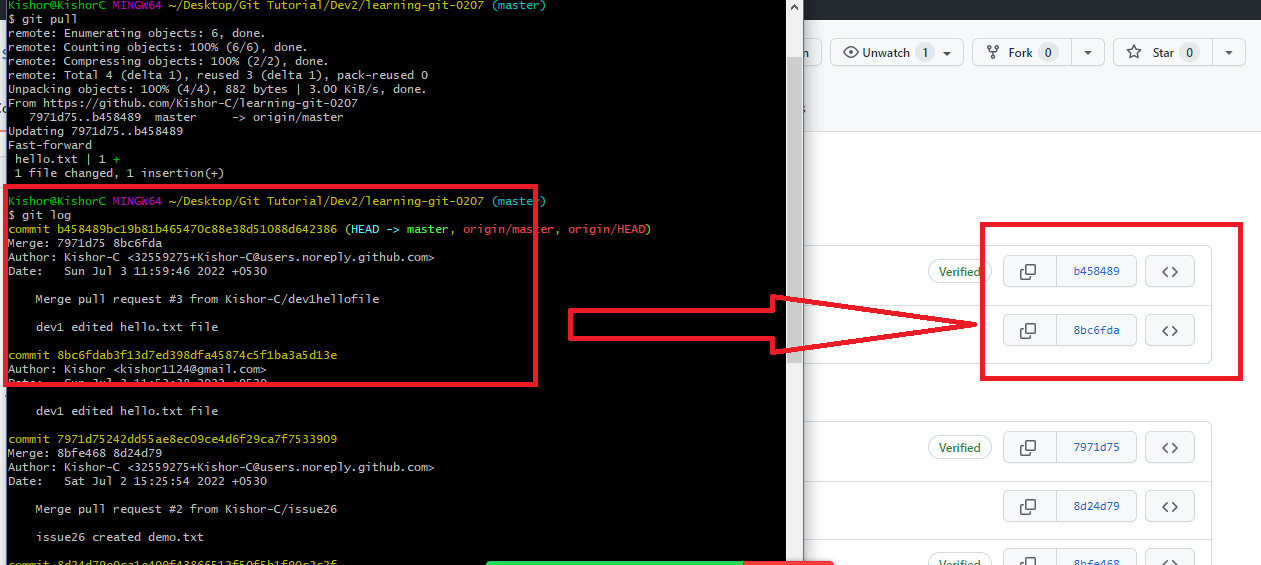
Why there was a conflict and what needs to be done?



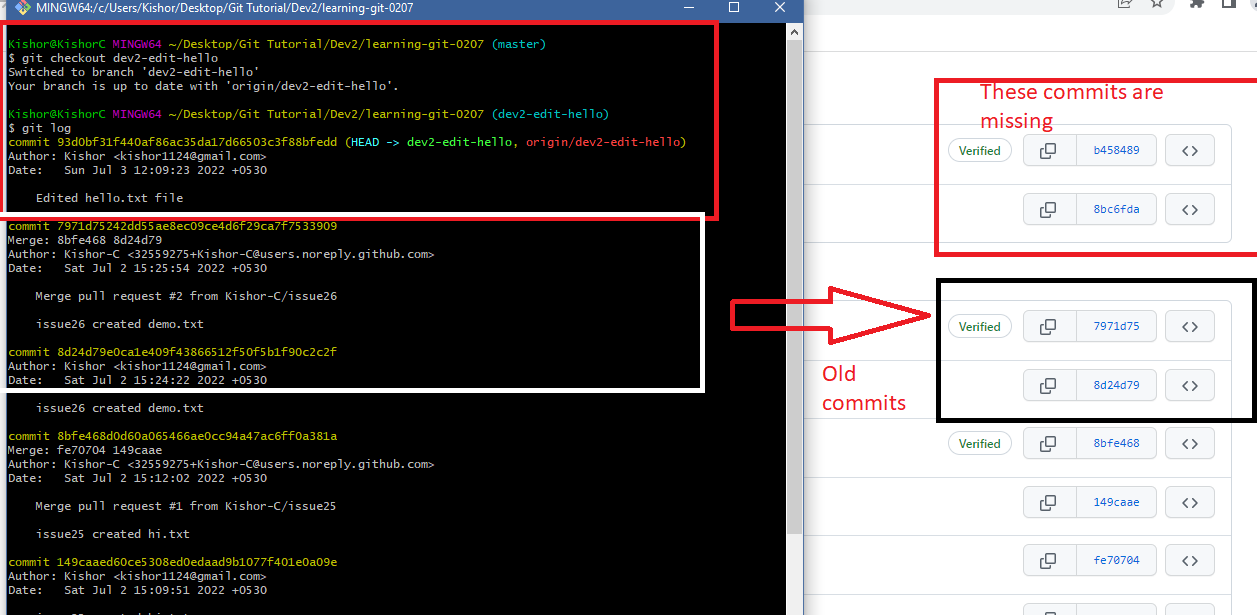
Firstly we need to checkout to local master & update the local master branch so that it will have the same versions of what remote master has then you can checkout to feature branch and merge the local master with feature branch



Once you use ‘git pull’ there was an automatic merge

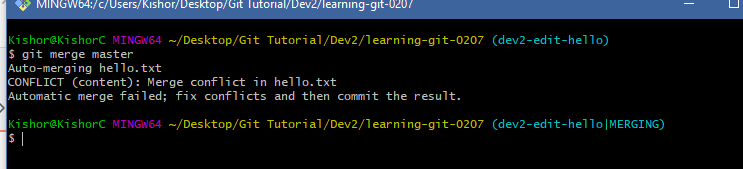


However the feature branch dev2-edit-hello is not up to date, we can see that using git log again in the feature branch



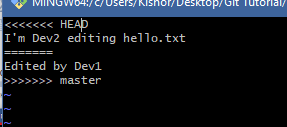
The remote has some versions which are not there in the feature branch, however local master is up to date

You can merge feature branch with local master since feature branch is not up to date with the local master you will still get merge conflict hence we must resolve it first with a new commit that will have all the missing versions of master branch in the feature branch.



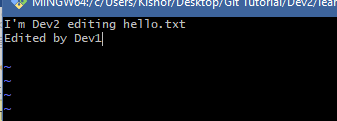
Note: ‘git merge master’ will merge the master branch history with dev2-edit-hello branch, it would’ve worked if feature branch was having same histories of local master since its missing we get the conflict error, the dev2-edit-hello|MERGING needs be resolved.

vi hello.txt: Since there is a conflict in the hello.txt it shows in the log

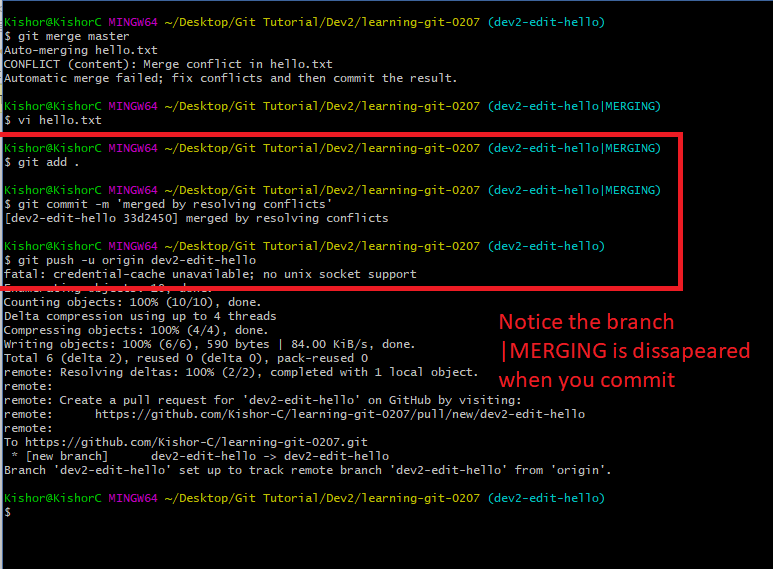


Here both Dev1 & Dev2 contents are merged and its in conflict you need to edit as per the requirement and create a new commit so that the new commit will have all the versions of the missing histories in the feature branch.

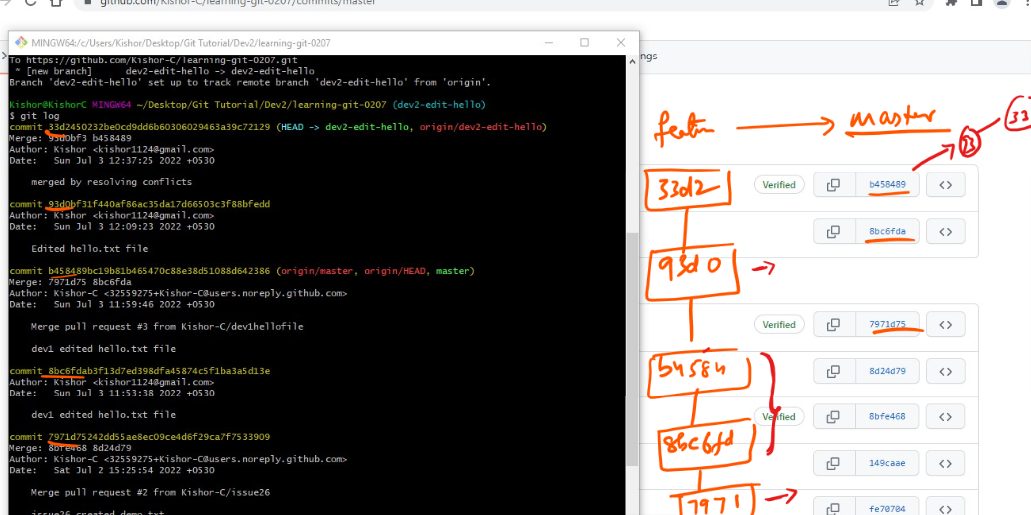
You can remove all the invalid contents & keep only valid contents



Then you can commit this change so that it will merge master branch & feature branch that will have all the missing histories in the feature branch.



Here the master had some commits & feature branch had some commits which were in merge conflict that was manually resolved and created a new commit.



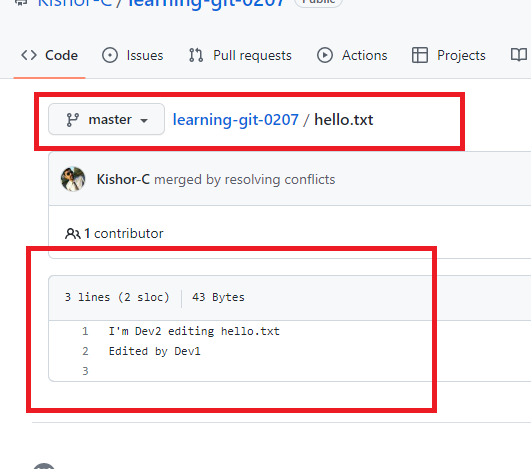
Since the feature branch is not missing b4584… & 8bc6fd… from the master branch it can be merge in the remote server without any problem.

Since we have pushed the feature branch in the remote we must see pull request

We can merge without any problem.

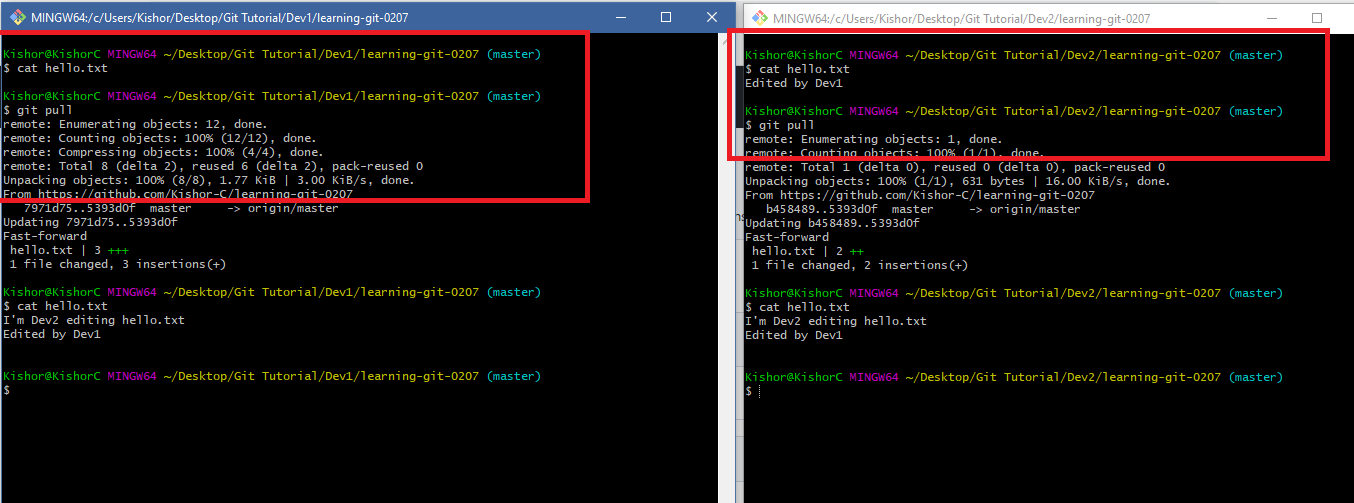


Earlier it made problem, but now there’s no problem, we can merge it successfully and delete the branch.



We are seeing the merge content of feature branch of Developer2 in the remote master, but there’s still something we need to do in both the developers machine.

Both the developers are still missing some commits compare to remote master.



In both the developers console we switched to master & used git pull so that it gets all the changes from the remote master that is up to date.

Note; Earlier both the local master was not having up to date changes of their remote master as the changes were done in the feature branch not in the local master branch

Git Organization:

It is a feature that makes multiple people to work in a team so that they can access the same repository to perform git operations like push, pull, merge, clone etc.

* As a member you can clone the project & work on it and update the remote repository

Activity:

* Clone the repository
* Create a feature branch and mention your name in the README.MD file
* Push the feature branch to the remote repository
* Admin (Kishor) will merge your branch

Summary of GIT:

* Git: It is distributed version controlling system
* Git Hub: It is an online Git repository website to store the remote repositories
* Git Stages: Working area, Staging area, Commit
* Git commands: git add, git commit, git push, git pull, git merge, git status, git log
* Git bash: A terminal to enter git commands
* Git Organization: It is a feature that helps you to work in a team so that group of people can collaborate their work in a single repository

HTML

Software’s required

* VSCode editor from <https://code.visualstudio.com/download>

Note: You can download the zip file & unzip it and use the application without installing

HTML: HTML stands for Hyper Text Markup Language, it is used to develop websites i.e., contents for your websites.

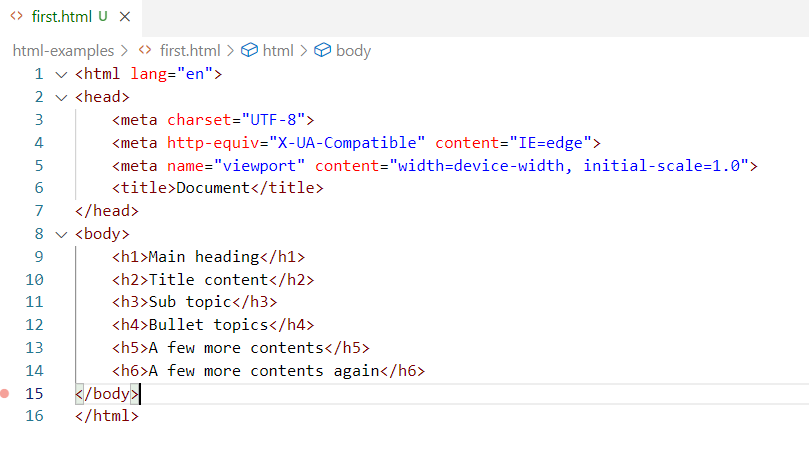
HTML is main language every browser understands, it understands other technologies as well like CSS & Javascript.

HTML has tags to render the contents, these tags are predefined which every browser understands, below are some useful tags

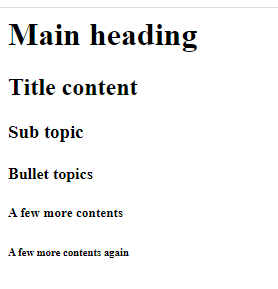
html, body, h1, h2, h3, h4, h5, h6, p, i, u, table, form, ol, ul, input, select, div, canvas, sup, sub, hr, br, img, link, script, …

HTML headings

There are 6 header tags to create heading with different size, they are h1, h2, h3, h4, h5, h6



Output:



Creating Images

You can add images using <img> tag it has attributes like src, width & height that gives information about the image

<img src = “url” width = “number” height = “number />

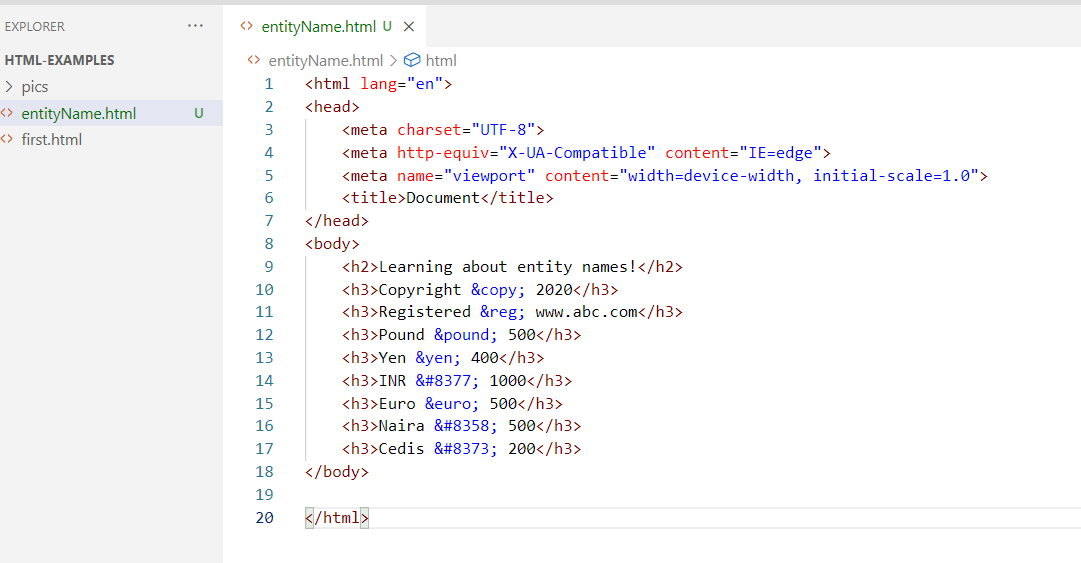
Note: <img> doesn’t need any closing with separate tag, it can be closed with the same tag as it is self closing tag

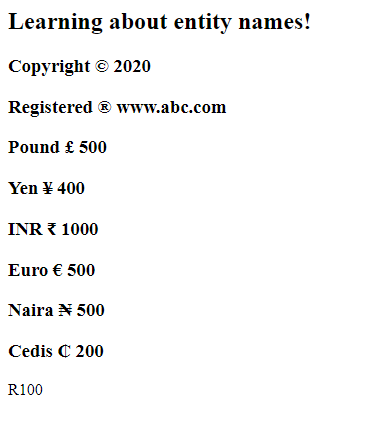


Output:



Entities: These are some special symbols that HTML can show in the browser, some have names & some have numbers

Output:



HTML Lists

HTML provides two types of lists to list the data

1. Ordered: This will have numbers
2. Unordered: This will not have numbers instead they will have bullets

Tag for ordered is <ol>

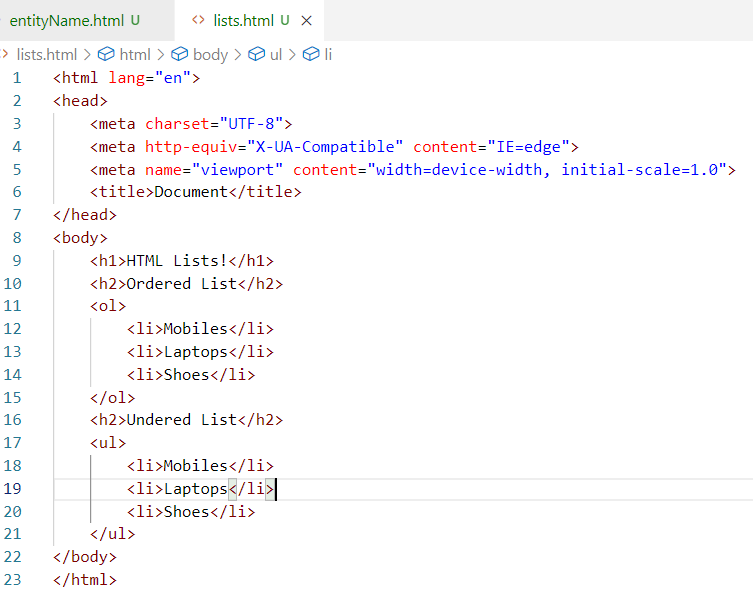
Tag for unordered is <ul>

<li> is the tag used to list the data, it can be inside <ol> or <ul>

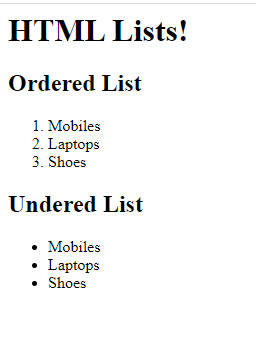
ol stands for ordered list

ul stands for un-ordered list

li stands for list item or list index



Output:



HTML tables

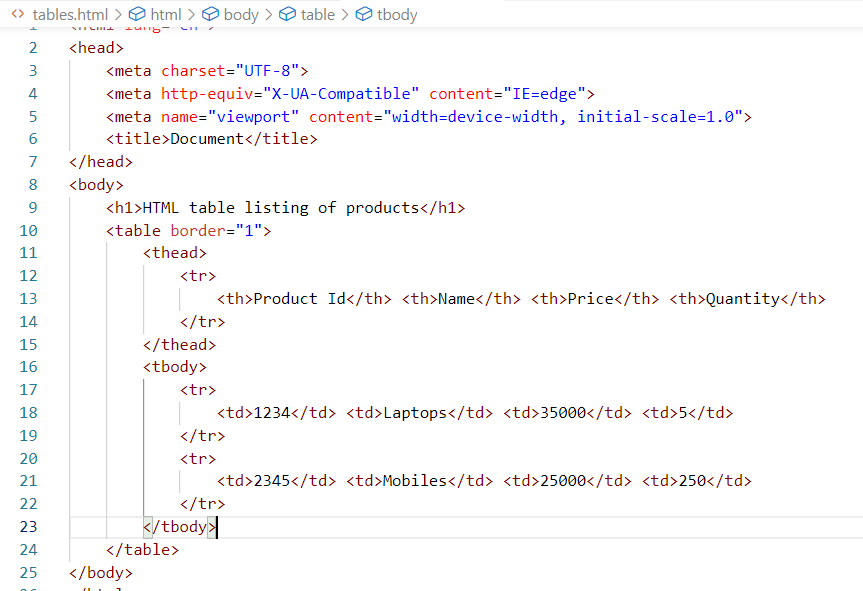
HTML provides <table> tag to create tables and it uses some tags to create rows & columns

<tr>: It is called as table row, used to create rows

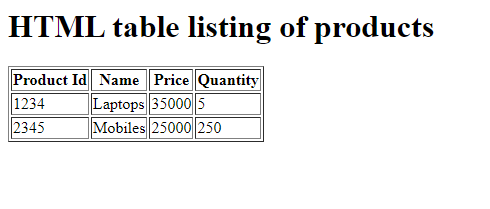
<th>: It is called as table heading, used to create columns for main row i.e., 1st row

<td>: It is called as table data, used to create columns for the rows

apart from these tags, you have <thead> & <tbody> they are used to create table headings & table body



Output:



HTML Forms

It helps us to enter data and submit to the backend application, in HTML you can only create form controls but backend code is written with some programming languages like Java, C#, Node.js and so on, so we must only create forms and enter the data for the time being.

<form>: It embeds different controls like input box, password box, radio button, checkbox, button, drop down and etc.

<form action = “url”>: The action is an extra attribute that can submit the data to the backend, but currently we don’t have any backend so we can just enter ‘#’, the ‘#’ is a symbol used to stay in the same page

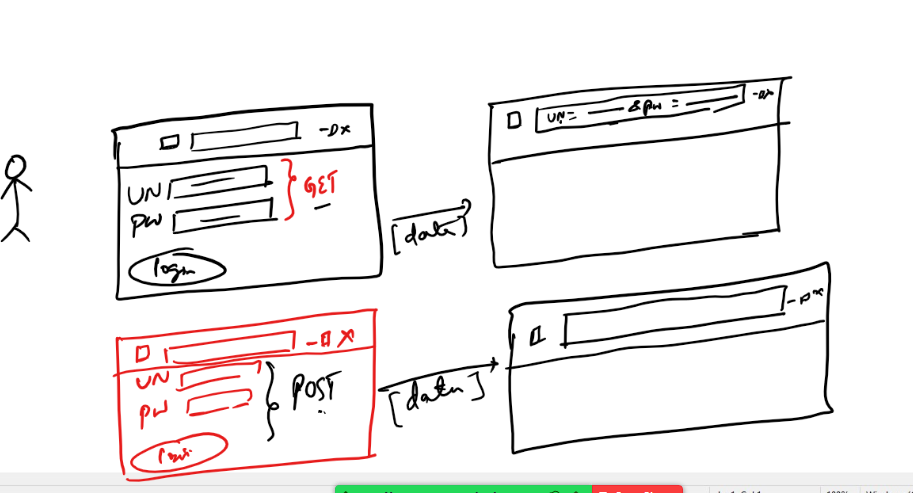
<form action = “#” method = “post”>:

Whenever you submit the form, the data inside the form goes to backend in two ways

1. In the URL of the request: This is when method = “GET”
2. In the Body of the request: This is when method = “POST”

What happens if the data goes in the URL: In that case the data will be shown in the URL of the browser, it means it will be visible

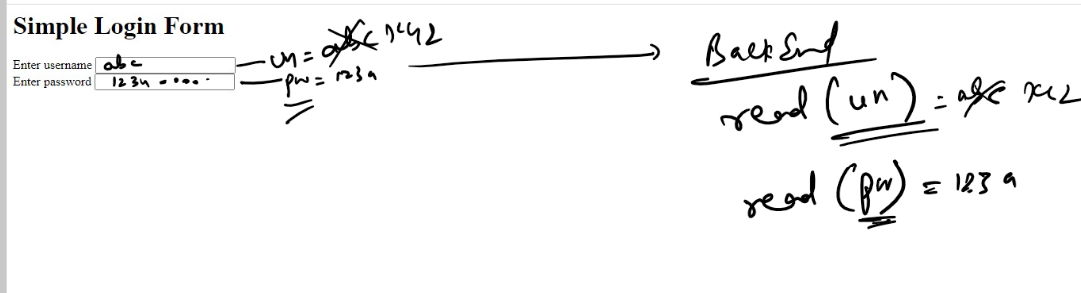
What happens if the data goes in the BODY: In that case the data wouldn’t be visible in the browser URL



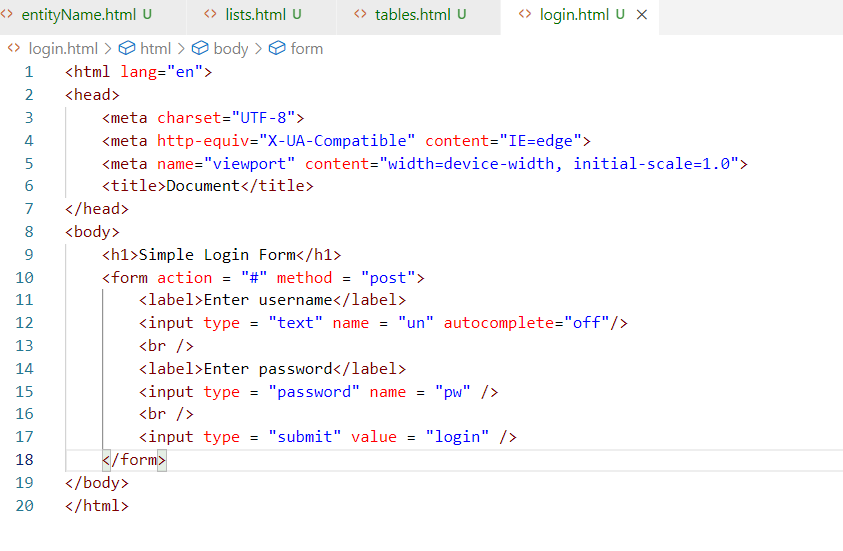
Note: If in case <form> doesn’t use method attribute then the form submits the data using method = “GET”, hence you better use method = “POST” explicitly.

<input type = “text” name = “un”>

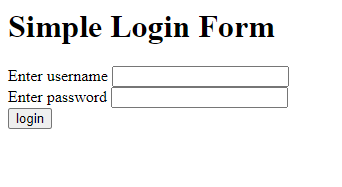
Here <input> creates an input box & type = “text” creates an input box that allows you to enter text, name = ‘un’ is the placeholder to store the value what user enters, so that backend programs can read(un) to get the value of un



Creating a simple login form

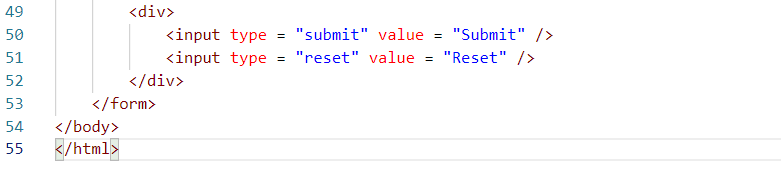


Output:



Other types of input controls like

radio, checkbox, dropdown, date etc.



Output:

