MERN Stack Training

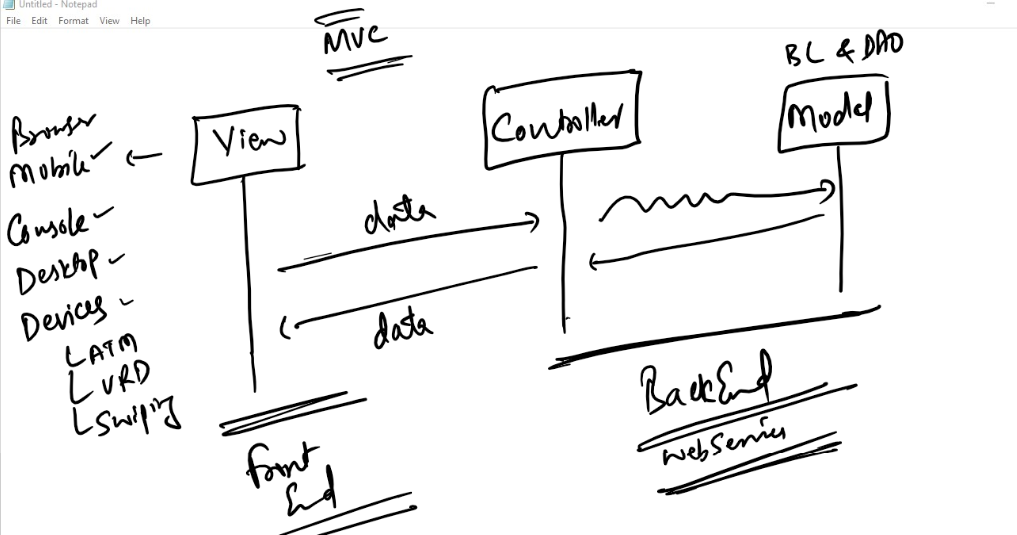
Contents

* Web Applications
* Fundamentals of Computer Programming Language
* Algorithms & Pseudocode
* Linux
* Git
* HTML, CSS, Javascript
* React.js, Redux, React Router
* Node.js
* Express.js
* MongoDB
* Testing
* Deployment

Application: It is a computer program which perform some tasks, there are two types of applications

1. Standalone application: These are the applications which you can use only by running on particular devices like mobile applications/desktop applications
2. Distributed application: These are the applications which you can access over the internet, there are two types of distribute applications
   1. Web applications
   2. Web services

Frontend & Backend



When you divide the applications into multiple layers like view, controller & model then you can have view in various types like mobiles, web browser, devices (Swiping machine, ATM Machine, …) and also backend can have webservices that helps to take data from any applications and return the data to any applications

Technologies used to create front-end & back-end applications

For Front-end

1. HTML, CSS, Javascript
2. React.js
3. Angular Framework

For Back-end

1. Javascript - Node.js & Express.js
2. Java - Spring Boot
3. Python
4. ASP.NET

Database: It is used by the backend applications to maintain the data, some of the database are: MongoDB, OracleDB, MySQL and so on

Front-end & Back-end together forms Fullstack development, a full stack development needs multiple technologies that needs to be connected

MERN stack: MongoDB, Express.js, React.js, Node.js

Evolution of Programming Languages

MLL: It is the language computers understand

C/C++: It is an high level languages but needs to be compiled to machine code, it is platform dependent

Java: It is also an high level language and its platform independent, it is more-preferred to write backend applications

JavaScript: It is also an high level language which is understood by browser & node.js, it can be used to develop both front-end & back-end applications

Before you write your first program you must know how to write instructions to the computers hence we need to have an idea on programming fundamentals like

1. Algorithms
2. Flowchart
3. Pseudocode

Note: These are not understood by any computers, these are for programmers to help writing code in different technologies like Javascript, C, C++, Java

Softwares required for MERN stack training

1. OS: Windows or Mac or Linux (UI-based)
2. RAM: Min 4GB
3. Privileges: Permission to download softwares/install them in your machine.
4. Virtual box - to learn linux
5. Git - to share your code with other team members in your company projects
6. VSCode - editor to write javascript code
7. Node.js - runtime environment to run javascript at the backend
8. MongoDB - database to maintain the data
9. Browser

Algorithm:

* It is a step by step procedure which are followed to get the desired result for the problem statement
* It is independent from any programming language
* It is not understood by any computer/compilers
* It can be written in any ways, there’s no any defined rules it must written in some standards
* It can be used to write programs which are understood by computers

Suppose you want to add two numbers

Step1: Start

Step2: Read number1

Step3: Read number2

Step4: Add number1 and number2 and store in a result

Step5: Print the result

Step6: End

Another way to write algorithms to add two numbers

Step1: Start

Step2: Take two numbers

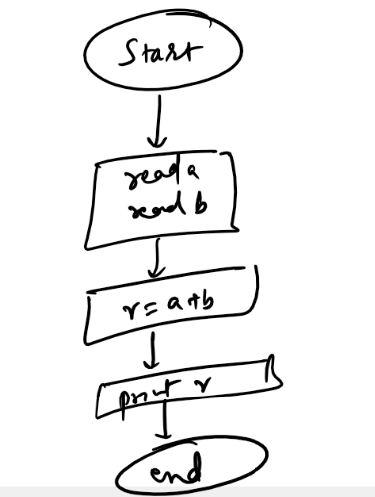
Step3: Add those two numbers and store in a result

Step4: Print result

Step5: End

Flowchart:

* It is a graphical notation of program sequence
* It is also language independent



Pseudocode:

It is also language independent, it can mimic the programming language by specifying programming constructs that needs to be used to solve problem

Note: This is also not understood by any computers

Ex: Pseudocode to add two numbers

Read x

Read y

r = x + y

print r

Pseudocode to find maximum of two numbers

Read x

Read y

if x > y then x is maximum

else y is maximum

Writing the actual programs in various languages

C program to add two numbers

void add() {   
 int x, y, z  
 printf(“Read x”);  
 scanf(“%d”, &x);  
 printf(“Read y”);  
 scanf(“%d”, &y);  
 z = x + y;  
 printf(“result = %d”, z);  
}

Java program to add two numbers

void add() {   
 int x, y, z;  
 System.out.println(“Read x”);  
 x = scanner.nextInt();   
 System.out.println(“Read y”);  
 y = scanner.nextInt();  
 z = x + y;  
 System.out.println(“result = “+z);  
}

Javascript program to add two numbers

x = prompt(“Read x”);  
y = prompt(“Read y”);

z = x + y;  
alert(z); // shows the result in dialogbox  
console.log(z);

Linux

It is a popular server OS used to deploy high end software’s, business oriented applications

* it is very powerful and faster than any UI based OS
* It is open source, it means any body can customize the Linux OS as per their need

What we must know from Linux

We must understand how to use linux commands either to open a file, write a file, open a folder, navigate to different path, install software’s

Important commands of Linux

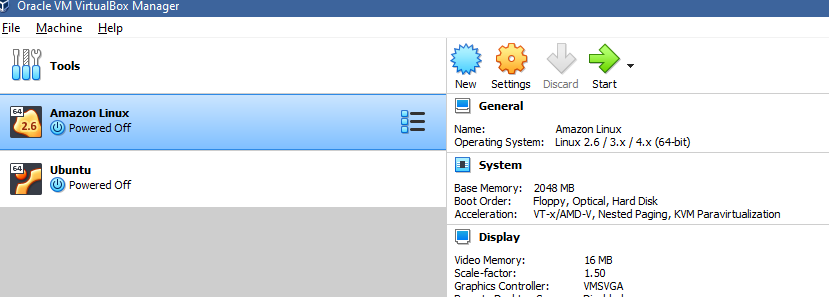
* ls
* pwd
* cat
* date
* cal
* touch
* rm
* cp
* cd

How to install linux in virtual box

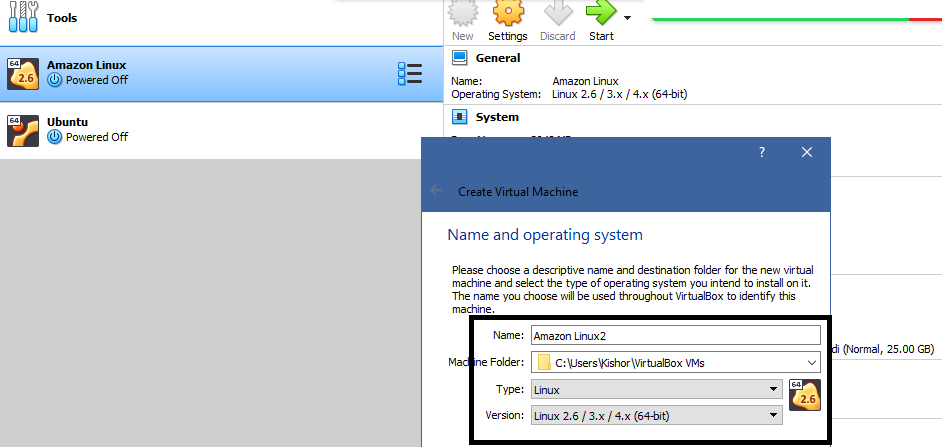
Follow this link

<https://www.how2shout.com/linux/download-amazon-linux-2-to-install-on-virtualbox/>

Step1: Open Virtual Box

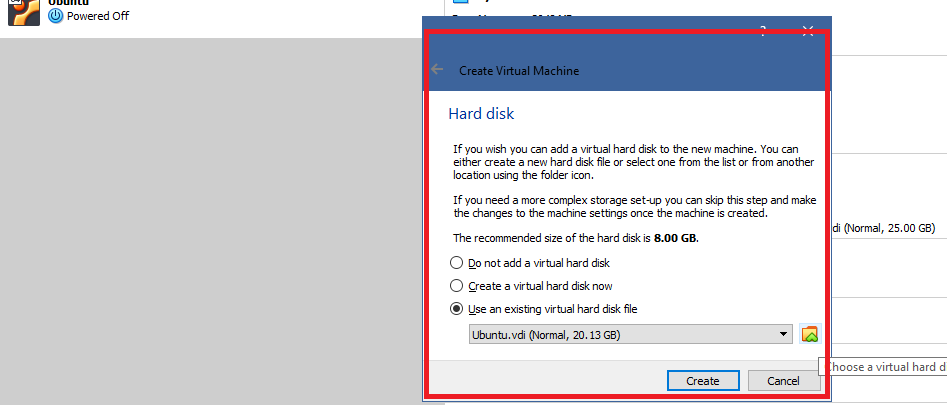


Step2: Click on New & Enter name & select type as Linux



Step3: Click Next & choose Ram size

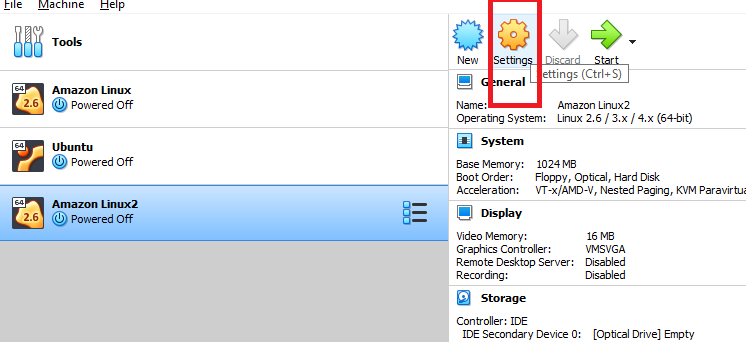
Step4: Select Use an existing virtual machine



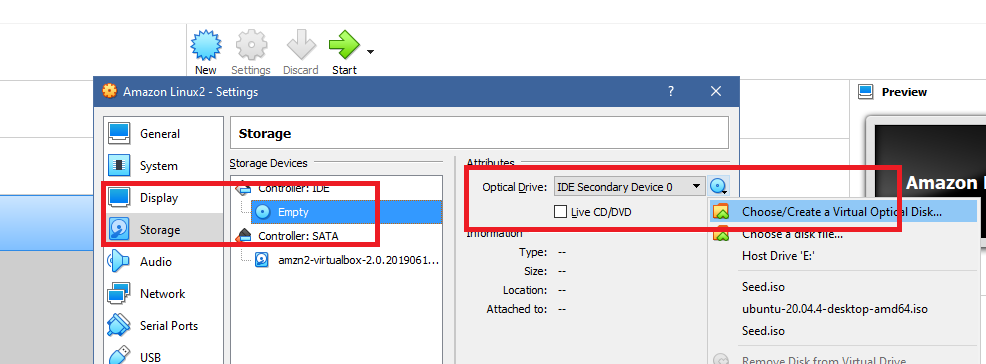
Choose virtual hard disk file, you will get an Add icon, select that you will need to browse and select the .vdi file which you have downloaded.

Step5: Click on Create

Step6: Select Settings



Step7: Choose Storage and Controller-> Empty, there you select the disc icon, where you will get choose/create a virtual optical disk



Step8: Click on Add and select seed.iso file

Step9: You can start the virtual machine, which will install linux on the virtual box

Note: Default username and password is

Username = ec2-user

Password = amazon

Note: When you enter password you may not see how many characters you are entering, just enter all the characters and hit enter key

Understanding Linux commands

pwd: Shows the present working directory

ls: Lists all the files & folders in the present working directory

mkdir: Used to create folder/directory

cd: To navigate from one folder to another folder

touch: To create files

cal: It shows the calendar

Steps:

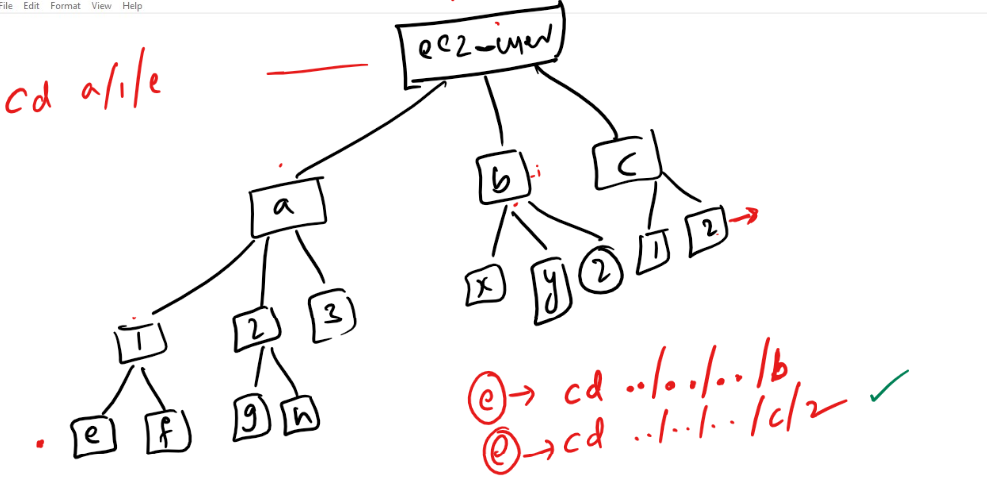
touch test.txt

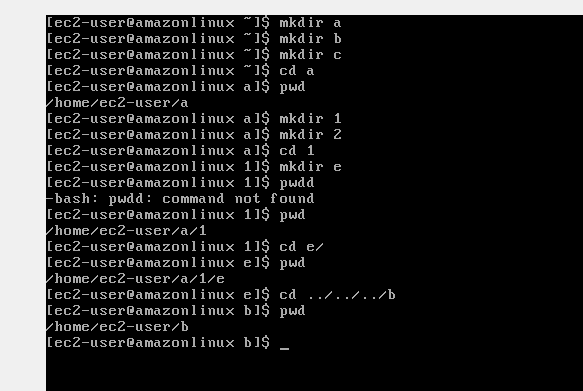
PRESS INS KEY

write some content

PRESS ESC KEY

Type :wq (Which appears at the bottom of the editor)





Day 3:

Agenda:

1. Linux grep and shell script
2. GIT

Grep: Global Regular Expression Print

It is mainly used to search the characters in the file and also you search various files & folders in your linux platform.

ex: grep pattern filename.txt



Shell Script

These are list of instructions that are stored in a .sh file, this simplifies user to keep frequently executing commands in one file and run it through script runner which is called as ‘bash’, this bash is present in /bin/sh location that can run any script files.

Note: You must begin every script file using #!/bin/sh so that the script will be run by the bash

Syntax of the script

echo “Some content”

The above line will print the expression you have written in the string

Creating variables

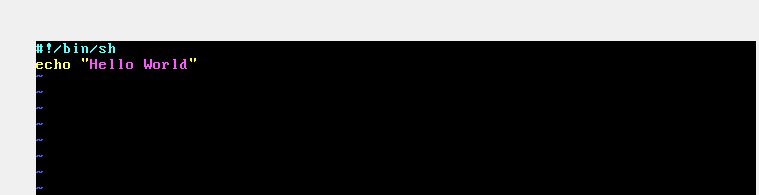
x = 20

y = 30

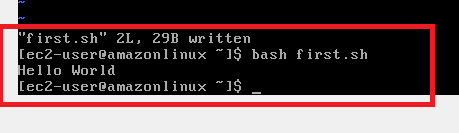
echo $x  
echo $y

The above command stores 20 to x & 30 to y and also prints their values.

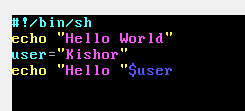
first.sh



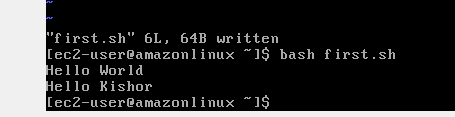
Use bash first.sh to execute the script



Storing the name in the variable

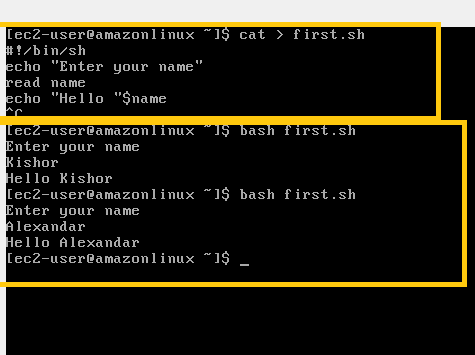


Output:



Reading the input from the user

syntax: read variable\_name



Calculating the data

#!/bin/sh

echo “Enter num1”

read num1

echo “Enter num2”

read num2

res = $(($num1 + $num2))

echo “Result = “$res

Output:

>> bash filename.sh

Summary

1. Linux commands
2. Grep
3. Shell Scripting

Git:

It is a distributed version controlling system or centralized version controlling system which keeps track everyone’s work through some version numbers that is internally generated by the git, the version number is an unique 40characters id that will help git to internally identify the changes done by the user.

Git will also help in collaborating the work done by multiple users within the same team and also it will maintain everyone’s work in a remote repository that is available to all the users and the same work will be maintained in local repository which is available locally to each user in their local machine.

Repository:

It is a working directory that will have files & folders and git version controlling folder that will track the changes, there are 2 types of repositories

1. Remote repository: It is a working directory which is present in the git hub and accessible over the internet
2. Local Repository: It is a working directory which is present in the user local machine

How git can collaborate everyone’s work

Git gives lot of commands that you can enter through Git bash terminal, some of the useful commands are add, commit, pull, push, merge, diff, log and so on.

How to download the remote repository in our local machine

Git gives a command called clone that needs to be used with the url of the remote repository

i.e., git clone remote-url

Note: The above command must be entered through git bash and also ensure the git bash opens in a folder that you can easily remember

ex:

git clone <https://github.com/Kishor-C/git-demo-2310.git>

How to update the local repository changes to the remote repository

git add filename [or] git add .

git commit -m ‘some message’

git push -u origin master [or] git push -u origin main

Note: Git throws author identity unknown on the very first time when recently installed git, hence you need to set user.name & user.email

Run below two commands

git config --global user.email “yourmail-id@domain.com”

git config --global user.name “your name”

ex:

git config --global user.email “kishor1124@gmail.com”  
git config --global user.name “Kishor-C”

Note: After this enter git commit -m ‘some message’

Summary of basic git clone & push

1. Created Remote repository
2. Cloned Remote repository: ‘git clone <<remote-url>>’
3. Created a file: ‘touch 1.txt’
4. Added to the staging area: ‘git add 1.txt’
5. Saved the changes: ‘git commit -m ‘some message’ [you need to configure user.name & user.email for the first time]
6. Push the changes: ‘git push -u origin main[or]master’ [you need to authorize the permission with git credential manager for the first time]

Branch: It is a pointer that will have history of commits done by the user, by default git gives a main branch which is also called as master branch, but we can also create custom branch which is called as feature branch

How to see the commits done by the user

git log is the command used in the local machine to see the commits done by various users

Activity:

create another file with the name 2.txt, then push it to the remote, ensure you are adding the changes to the staging area, commit and then push

Git cycle

Working Directory -> Staging Area -> Commit -> Push

Working Directory will have the changes done

Staging Area will have the changes that needs to be tracked

Commit will save all the changes in the staging area

Push will push the changes to the remote that are committed.

Some commands list

git status: It shows the changes which are tracked & untracked in the staging area

git add . : it adds all the changes to the staging area

git add file/directory: It adds the file/directory to the staging area

git log: It lists all the commits of a branch

git commit -m ‘some message’: it commits the changes tracked in the staging area.

Git Fork  
It is used to create a clone of a repository from others account in your account, the repository created in your account will be completely independent

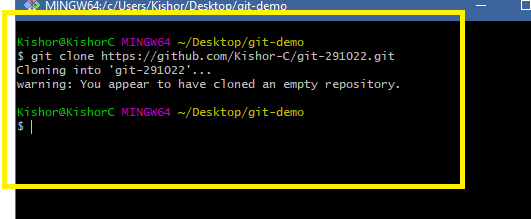
Activity

1. Clone the forked repository from your account to your local machine in a new folder which is not part of your old local repository
2. Create some files and push those changes to the remote
3. Once Done let me know

Summary of Git process

1. Firstly create a remote repository
2. Clone that remote repository in your local machine

git clone url-of-repository



Note: Ensure you are not inside an existing repository while clone a new repository

1. Navigate to the git working directory first using ‘cd’ command

cd git-291022

1. Make some changes

touch hello.txt

1. Add the changes to the staging area

git add .

1. Commit the changes with a meaningful message

git commit -m ‘created hello file’

1. Push the changes to the remote

git push -u origin branch-name >> master/main/custom-branch

Apart from these commands there are some more useful commands like

1. git log --oneline
2. git status
3. git branch
4. git checkout
5. git merge

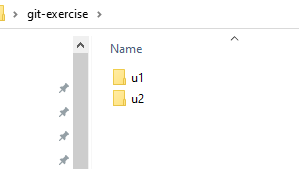
Exercise

Making changes in the same file by multiple users and updating in the remote repository

1. Create a new repository remotely with some name
2. Clone that repository in your local machine using `git clone remote-repo-url`
3. Navigate to the repository & create a new file and push that file to the remote
   1. touch 1.txt
   2. git add .
   3. git commit -m ‘create 1.txt’
   4. git push -u origin master [ or ] main

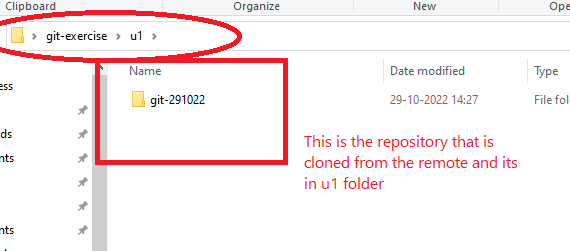
Note: Before creating 2 folders you must have atleast one commit hence you need to all the above steps

1. In local machine create 2 folders with the name u1 & u2 in a non-git repository

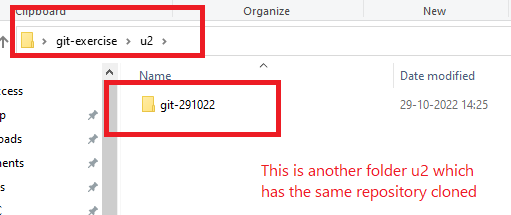


1. Ensure both the folders have the same repository

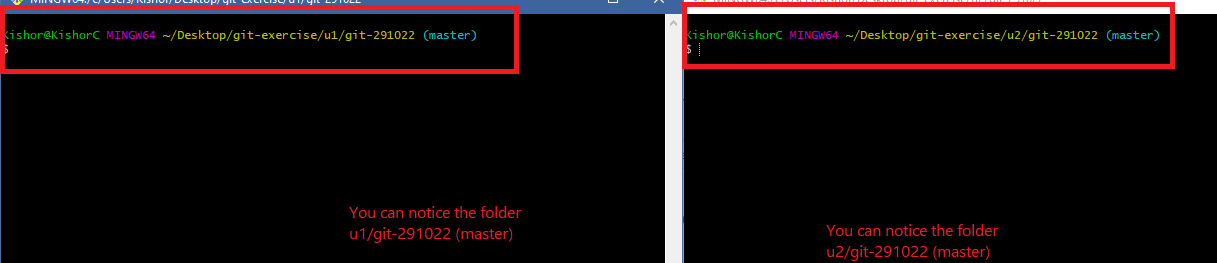
Repository in u1 >> git-291022



Repository in u2 >> git-291022

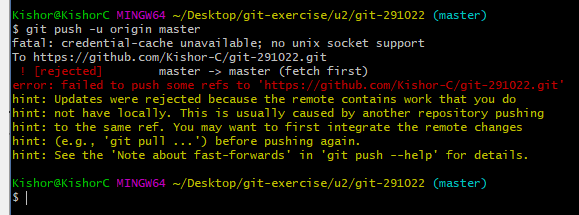


1. Open Git bash from both the folders repository

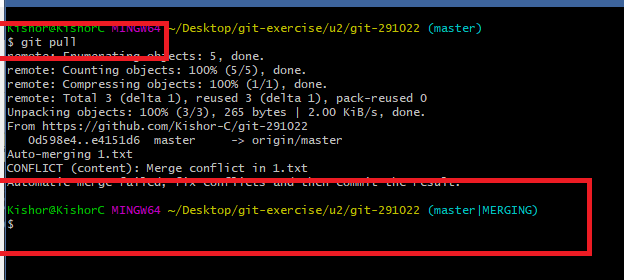


1. In u1 repository edit the file 1.txt and push that to the remote using below commands
   1. git add .
   2. git commit -m ‘modified 1.txt’
   3. git push -u origin master
2. In u2 repository edit the same file 1.txt and push that to the remote using below commands
   1. git add .
   2. git commit -m ‘modified 1.txt’
   3. git push -u origin master

Note: You will notice the push rejection

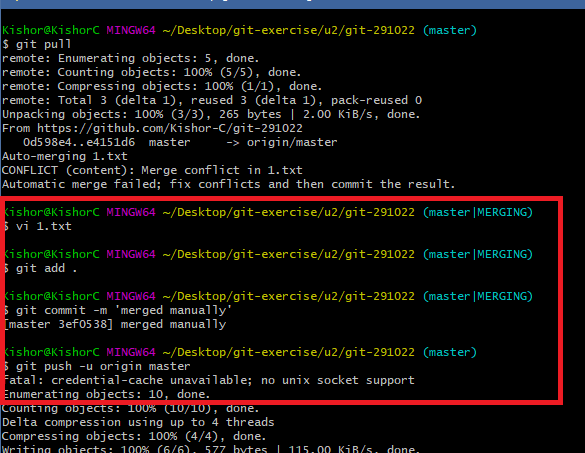


1. In u2 repository you must use `git pull`>> this will give a new branch saying auto-merge conflict



Use vi and edit the 1.txt where you will see some random characters

1. Once you edit 1.txt you need to re-add & re-commit the changes & push the work to the remote



1. Now you can see the remote & local repository uptodate
2. You can enter git pull in the u1 repository (ensure u1 has not done any new commits) to see the updates done by u2 repository

Rules we need to follow while working in the GIT

1. Never work in the master/main branch in the local repository
2. Any changes use wants do must be done in feature branch which is a copy of the master/main branch
3. Push only the feature branch from the local repository and pull the master/main branch from remote repository, it will always ensure that master/main branch in all the machines are clean and encourages the user at the server side to review everyone’s work
4. Someone at the remote repository can have a control over merging of branches, this user can also update others to resolve the merge conflicts if at the merge conflict occurs at the remote side
5. Local master must always get the changes from the Remote master
6. Local feature branch should be merged with the changes available in Local master after it pulls the changes from remote master

Note: It’s a good practice to delete the feature branch that is pushed in the remote repository, if not deleted then whoever pulls the remote will get all the branches present in the remote master, its okay if the feature branch is not deleted in the local repository, may be the use wish to work in the same branch

How to create a new branch

git branch branch-name

ex: git branch issue-runtime-error

ex: git branch branch1-user1

How to switch from one branch to another

git checkout branch-name

ex: git checkout issue-runtime-error

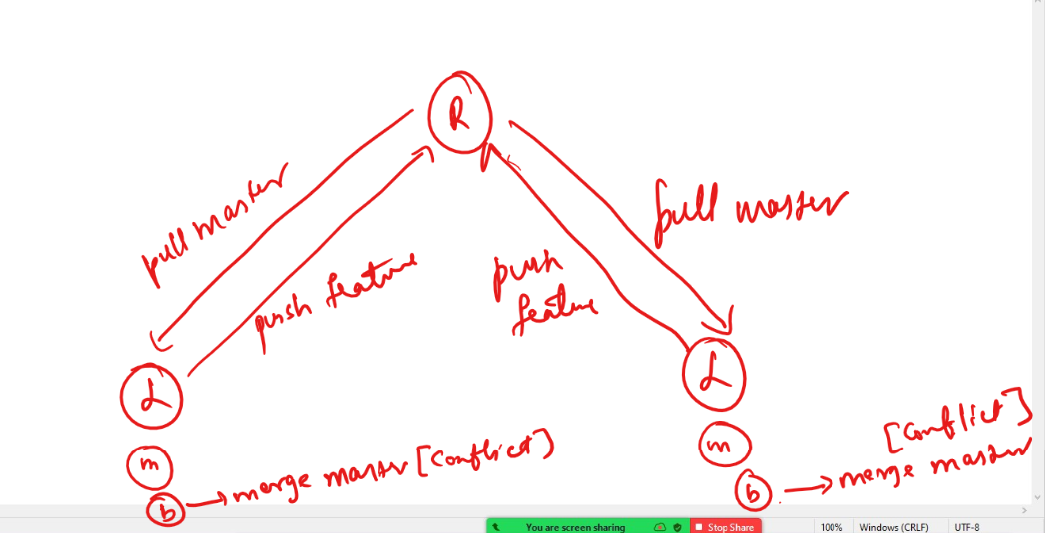
ex: git checkout branch1-user1

How to push the feature branch

git push -u origin branch-name

How to merge the local master with the feature branch

git merge master [or] main



Summary of how to work in GIT

1. Create feature branch & make changes in the feature branch and push the feature branch to the remote
2. If there’s a merge conflict in the remote then pull the remote to the local master and checkout to the feature branch, then merge the local master with the feature branch & then push the feature branch

List of commands to follow

1. git branch b1
2. git checkout b1
3. git add
4. git commit
5. git push -u origin b1
6. Go to git-hub and merge the b1 with remote master
   1. If merge is success then delete the b1 in the remote
   2. If merge fails then also delete the b1 in the remote, but in the local pull the remote master and checkout to the b1 and merge the local master with b1 using ‘git merge master’

Note: While using git pull, checkout to master/main using ‘git checkout master’ or ‘git checkout main’

Activity

1. In local repository create a feature branch with the name b1 for user1 and another branch b2 for user2
2. Make changes in each branch and push that branch to the remote & merge the branch in remote, use the command git push -u origin branch-name to push the branch to the remote
3. Try to edit the same file in both the feature branches and push the changes to the remote and observe the merge conflict you get.
4. If you get the merge conflict use ‘git pull’ in master branch & checkout to the feature branch and use ‘git merge master’ or ‘git merge main’, then resolve the merge conflict by editing the file and push the changes to the remote

Use the following commands to perform these activity

branch-creation: git branch branch-name

checkout: git checkout branch-name

.gitignore:

It will have list of files & folders that are not tracked if changes are done

HTML

Software required:

* Visual Studio Code

<https://code.visualstudio.com/>