**DATA OPS (Development & Operation)**

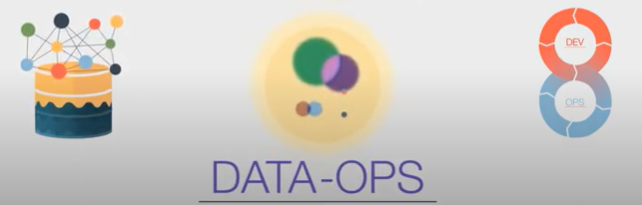
**Introduction:**

**Data Ops** is an automated, process-oriented methodology, used by analytic and data teams, to improve the quality and reduce the cycle time of data analytics. Data Ops began as a set of best practices, it has become a new and independent approach to data analytics.

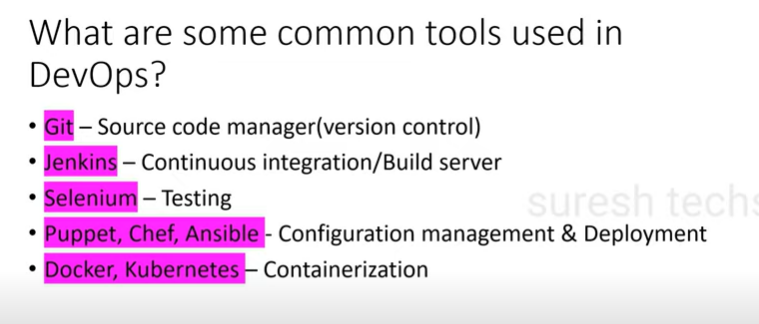
**DevOps** is a set of practices that combines software development (Dev) and IT operation (Ops) team to automate the project, Dev Ops is a Continuous Integration & Continuous Delivery.

Data Opps is combinations of Analysist, Data Science, Developers and Operations participating together entire lifecycle

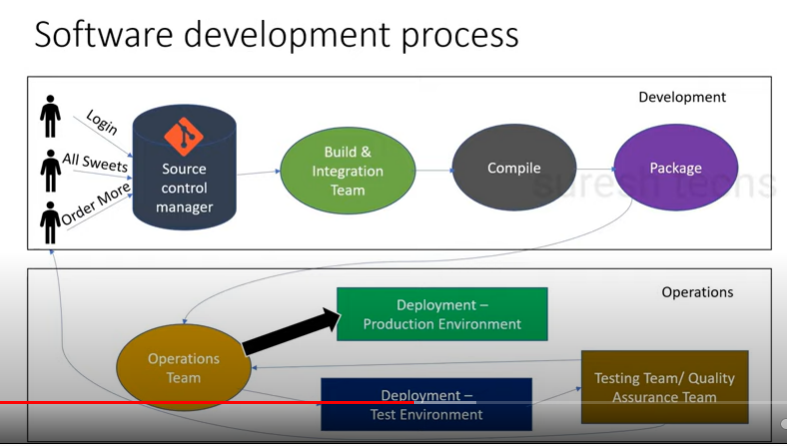


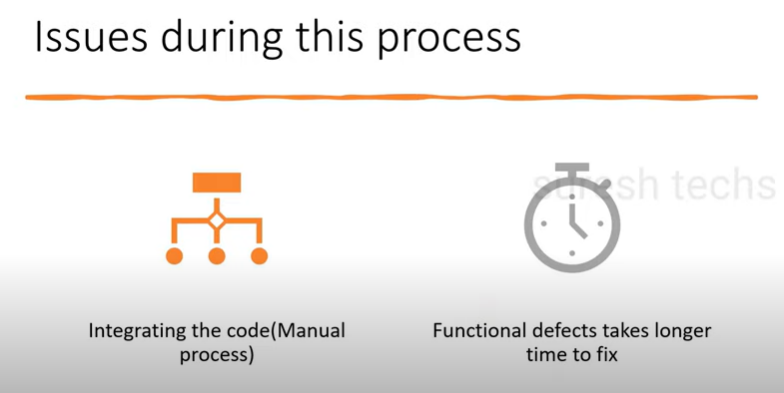


**DevOps Tools**



**Traditional System development**

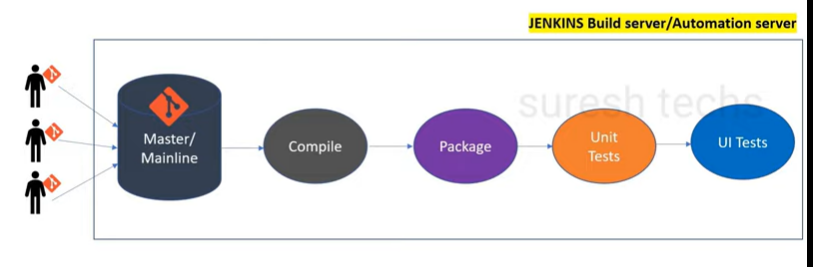


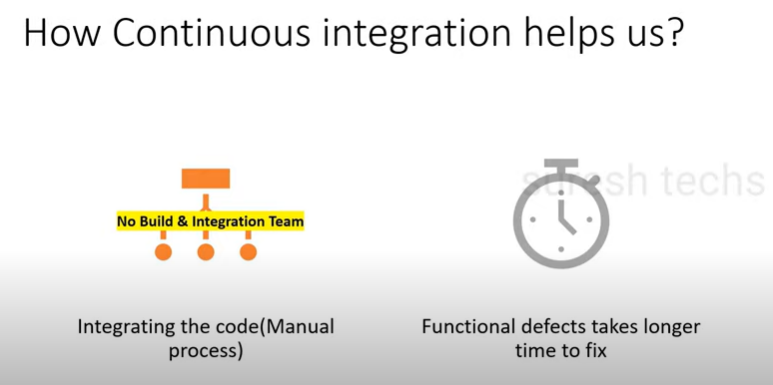


**Dev Ops is a Continuous Integration & Continuous Delivery**

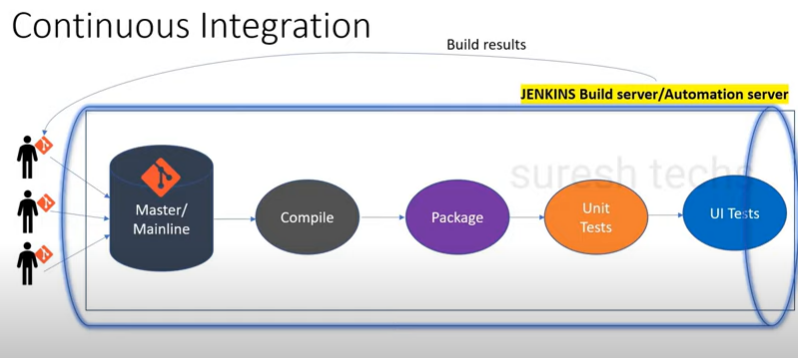
Junkins (Build server): Its Automation server

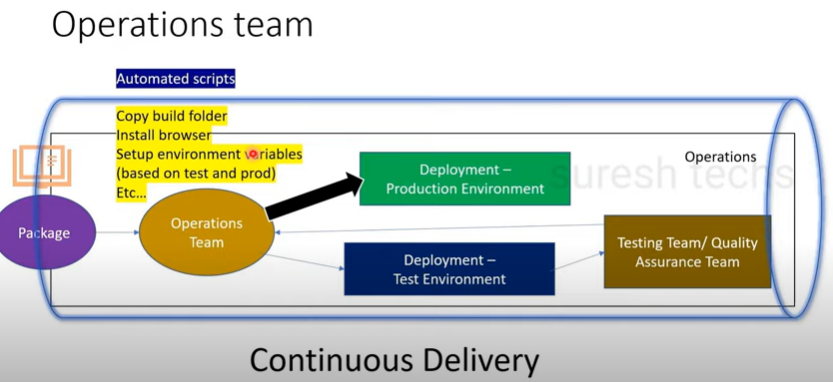
Its Automate the parts of software development related to Building, testing and deploying





Pipeline





Software development life cycle

1.Plan

2.requirements

3.Design

4.Development

5.Testing

6. Deploying /Deliver

**WHAT IS GIT**

Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

Git has been designed with the integrity of managed source code as a top priority, It has the functionality, performance, security and flexibility that most teams and individual developers need.

**Why/Who should use Git:** To Manage complex projects and code.

**Features**

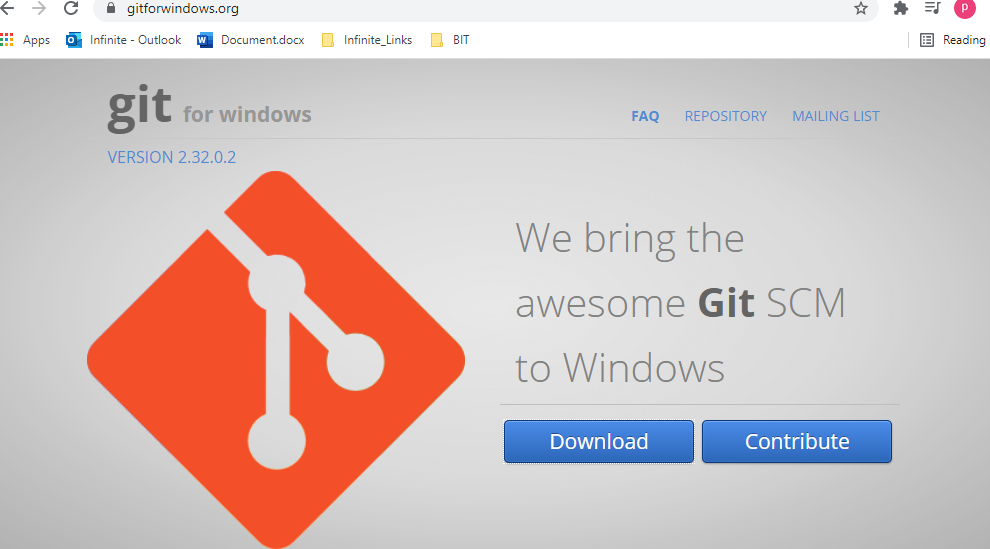
Distributed source control system

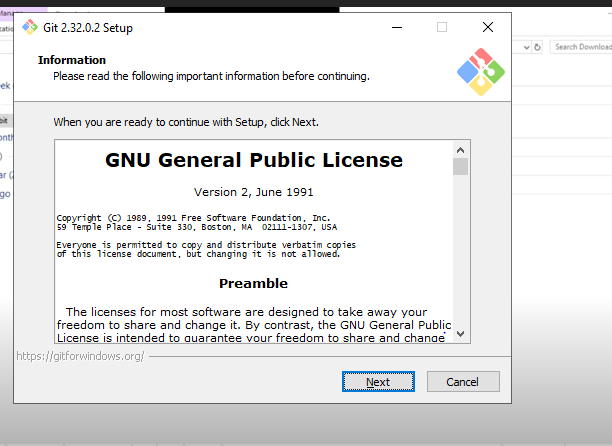
Open source

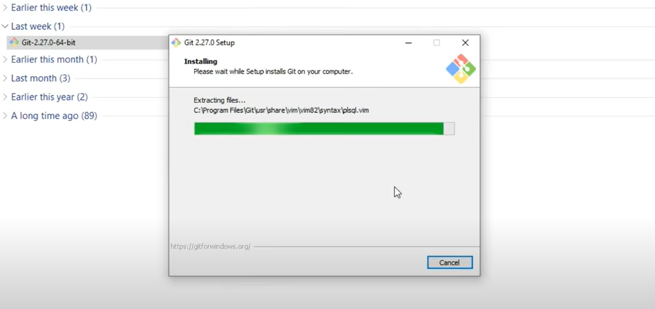
Large and active community

**How to Install GIT**

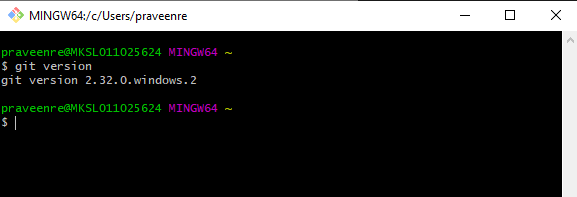
<https://gitforwindows.org/>







To check Git Version



**Handling Projects**



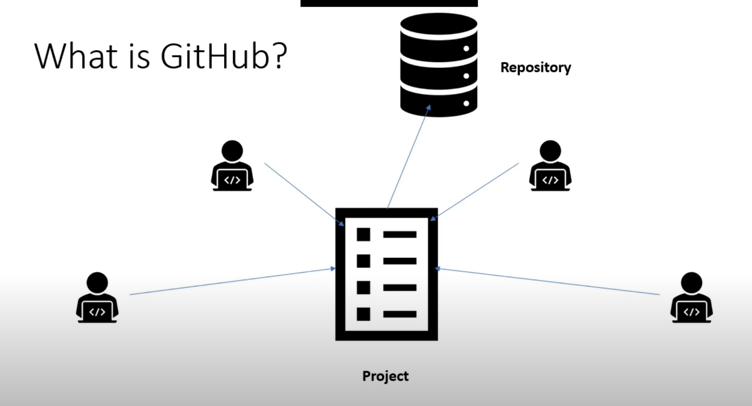
Repository exists in



We can place repositories in GIT HUB or GIT LAB

**What is GIT HUB**

**Github** is a web-based platform **used** for version control. Git simplifies the process of working with other people and makes it easy to collaborate on projects. Team members can work on files and easily merge their changes in with the **master** branch of the project.

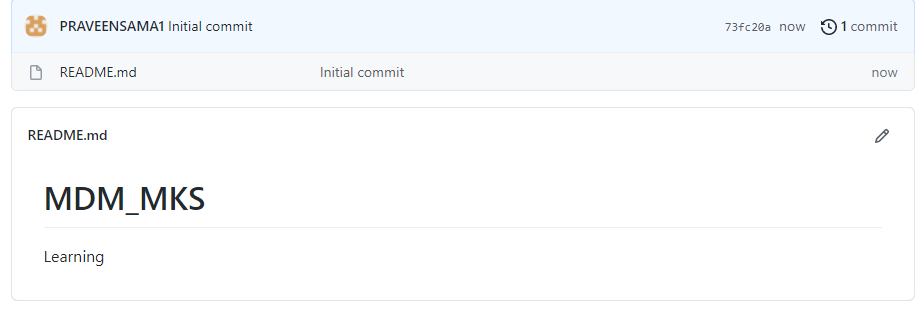




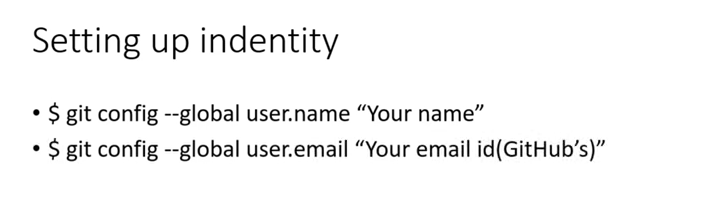
**CREATING GITHUB ACCOUNT**

Signup: <https://github.com>

Create Repository: <https://github.com/join/get-started>



**CONFIGURE GIT**



**Global : Personal information (gitconfig file)**

**System: All users information**

**Local: Repository information**

**commands**

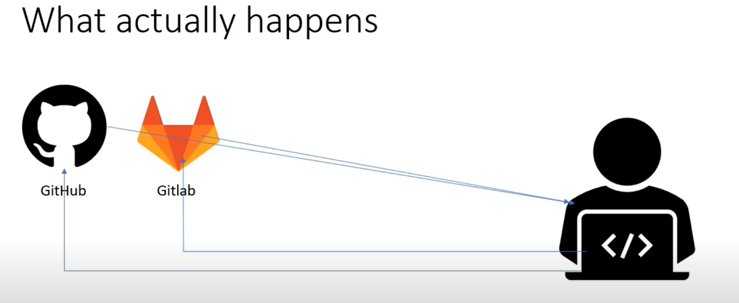
**$ git config –global**

**$ git config –global –e**

**praveenre@MKSL011025624 MINGW64 ~**

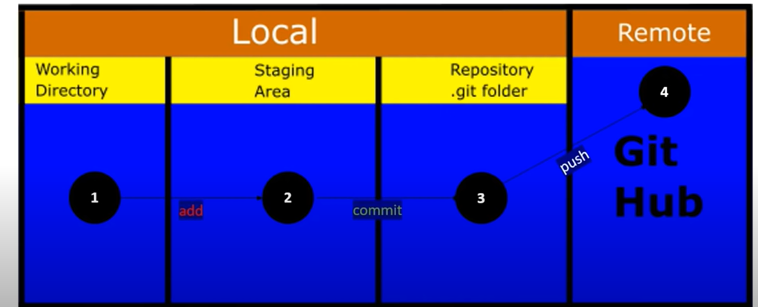
**$ git config --global –e**

**GIT WORKFLOW**



**Basic Terms**

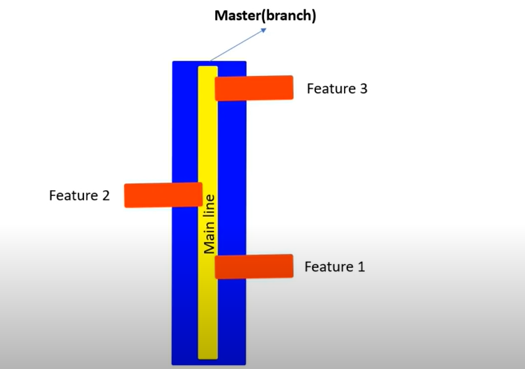
* **Commit: History info**
* **Clone: Exact copy**
* **Tracking/ untracking files: Tracking information of files and folders.**
* **Branch**



**Master:** The default branch name in Git is master. As you start making commits, you're given a master branch that points to the last commit you made.

**Origin: Server default name**

The term "**git origin master**" is used in the context of a remote repository. It is used to deal with the remote repository. The term **origin** comes from where repository original situated and **master** stands for the main branch.

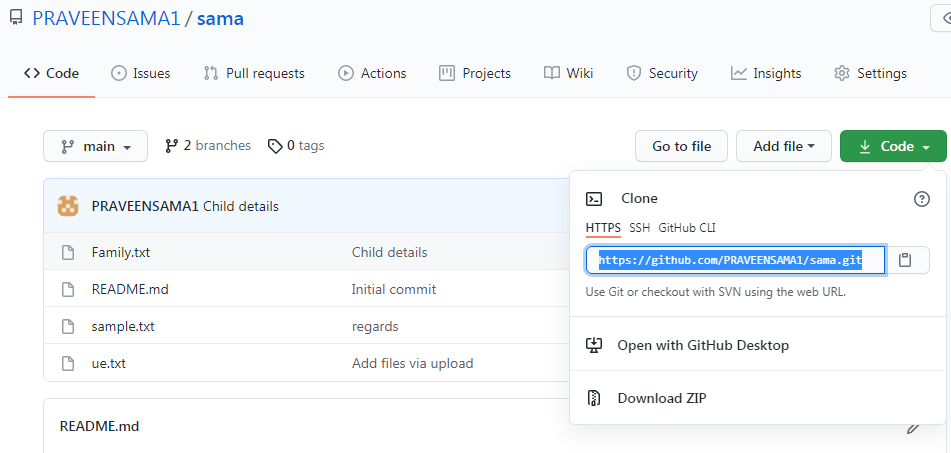


**CLONING EXISTING REPO**

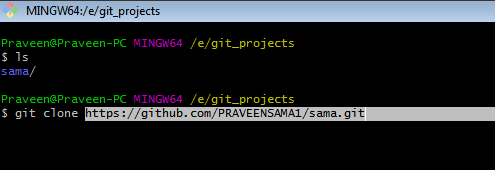
GIT CLONE <REPO-URL>

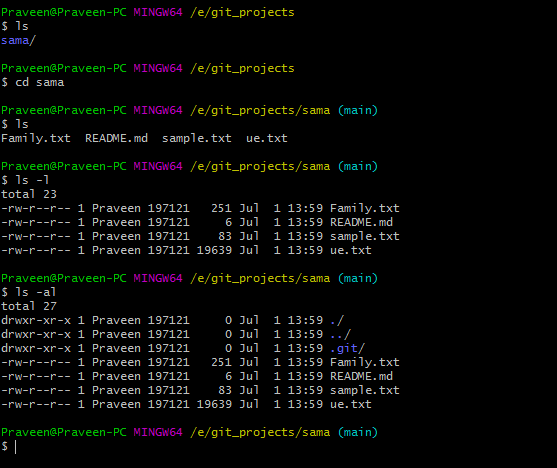
<https://github.com/PRAVEENSAMA1/sama>

git clone <repo-url>

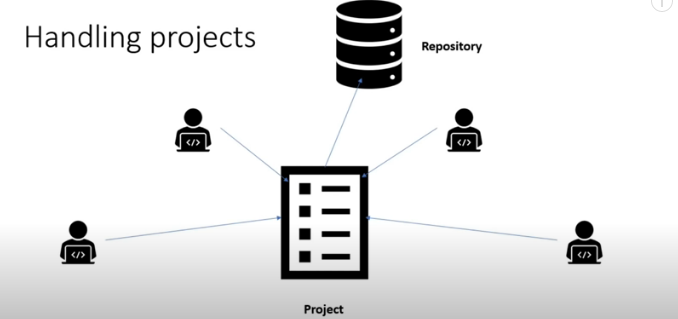


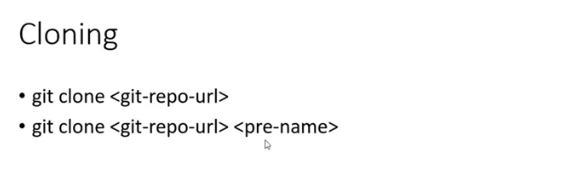
cloning projects





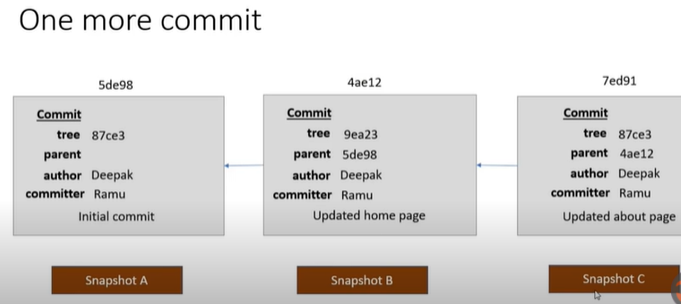
Cloning

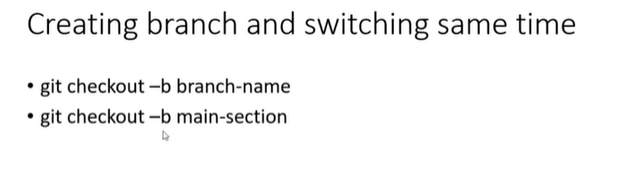


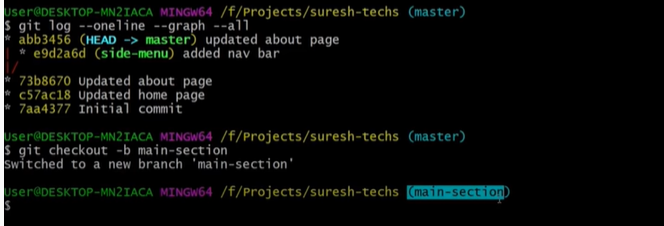




Commit



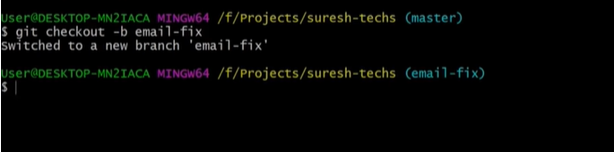


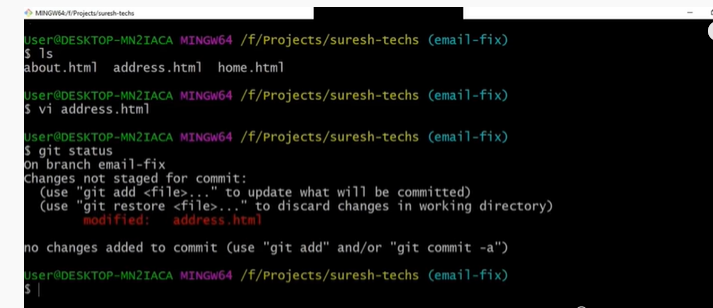


Git log --oneline --graph –all

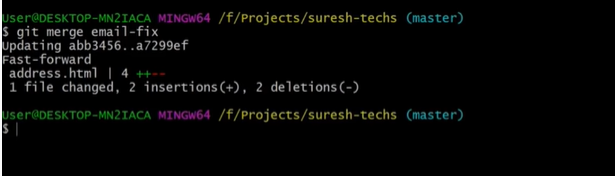
Create New branch

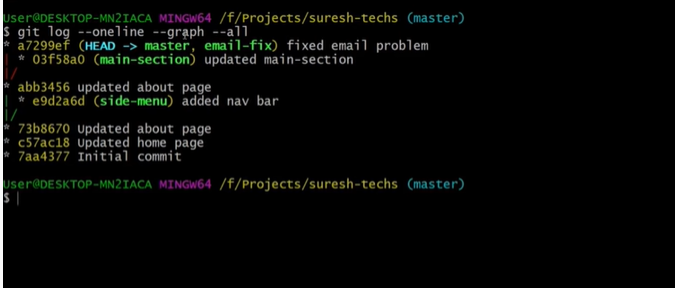
Git checkout -b email –fix





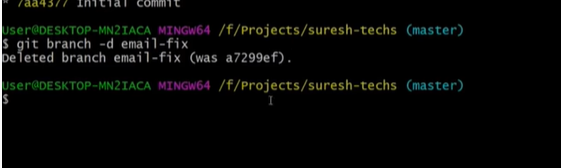
**Merging**

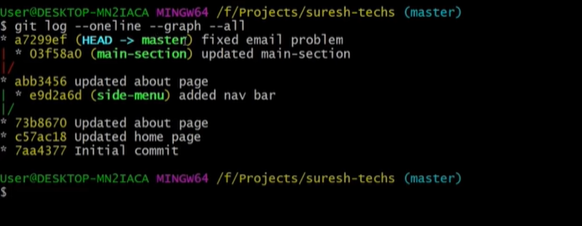




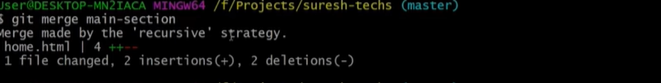
Deleting branch

Git branch –d email –fix





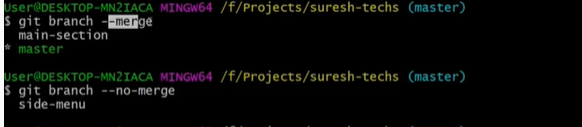
Merge with recursive strategy



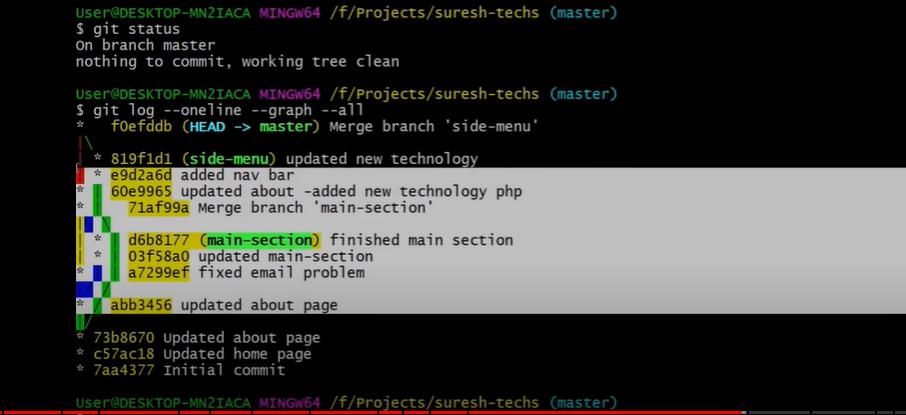
Merge conflicts

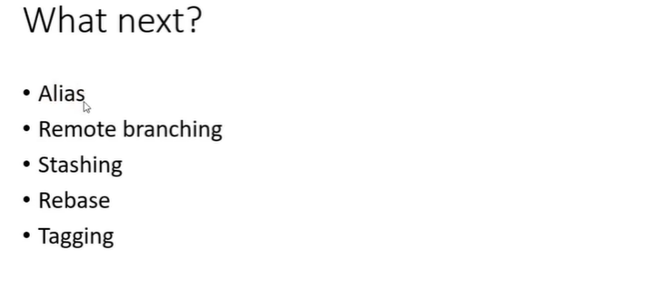
Git branch --Merge

Git branch --No-merge

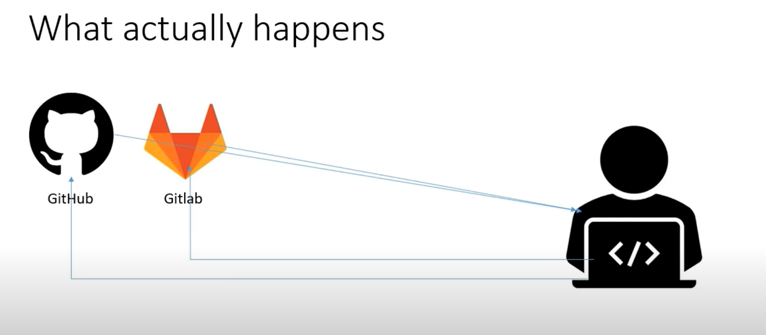


Conflict Merging





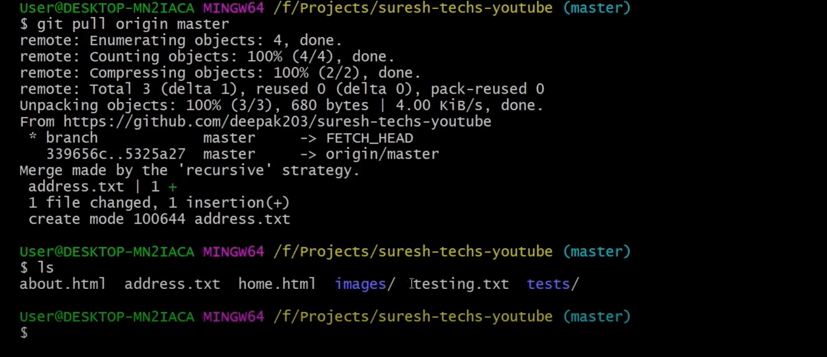
**Remote Branching**



FETCH & PULL

GIT FETCH ORGIN MASTER

GIT PULL ORIGIN MASTER



Get log –oneline –all –graph

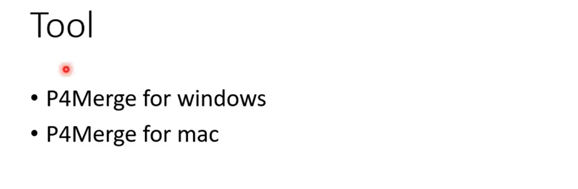
Get log --oneline --all –graph

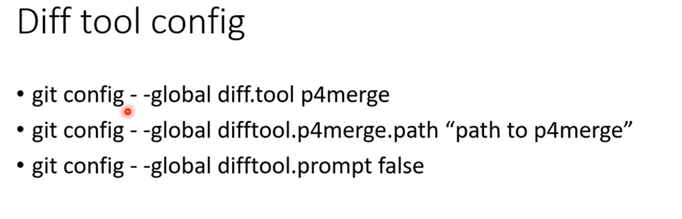
Get pull origin master

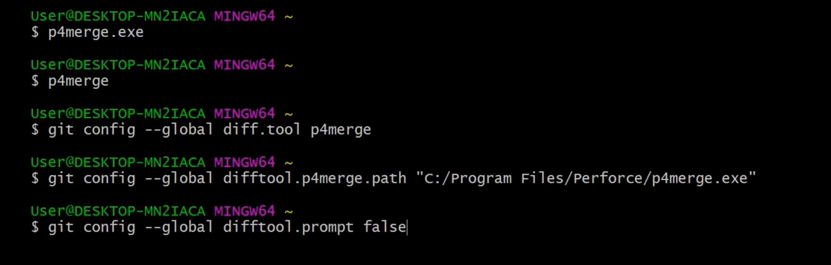
Get push origin master

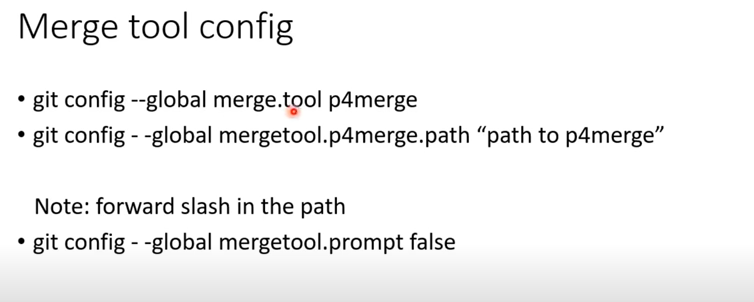
P4MErge





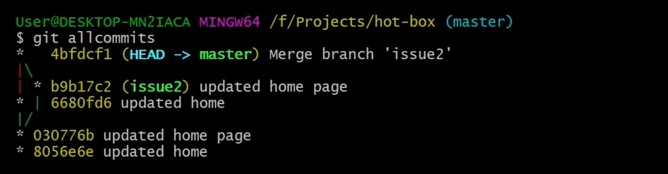






Alias

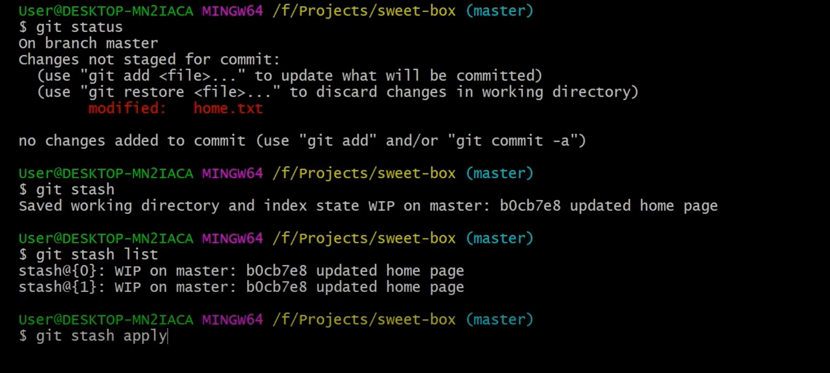


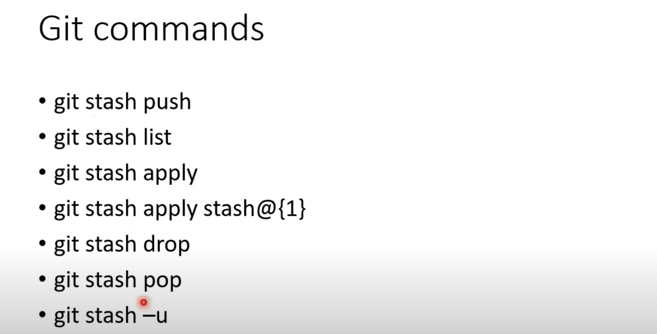


Git Rebase



Stashing (recycle bin )





**Git branch from Stash**

**New branch creating Git shash**

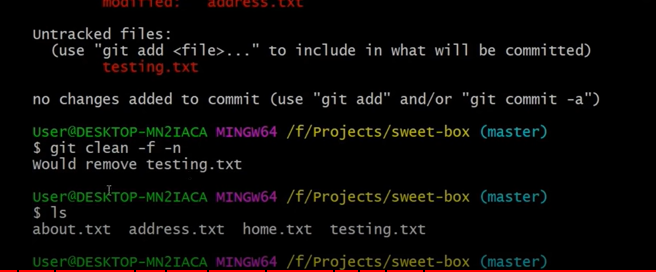
**Git shash branch \*\*\*\*\***



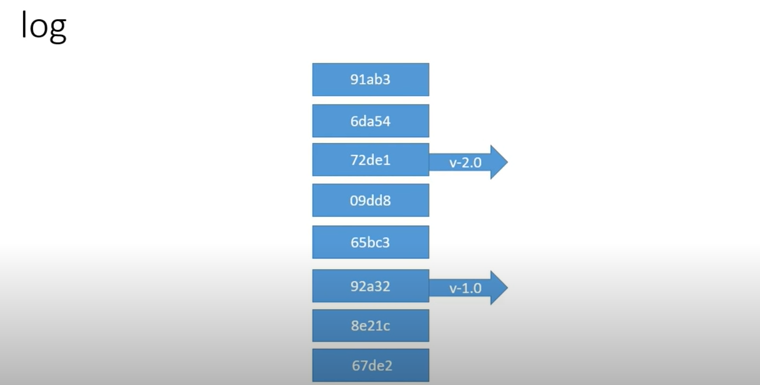
GIT CLEANING

Always deleted untracked files

DRY RUN



GIT TAGGING



Git tag -list

Git tag v1.0