**Git Version Control**

**Index**

**Introduction To Version Control**

**Why We Use Version Control**

**Types Of Version Control**

**What is Git**

**Creating Repository**

**File System Of Git**

**Adding Files To Project**

**Status Of Project**

**Committing**

**Ignore Certain File System**

**Git Logs**

**Cloning Repository**

**Branching**

**Merging**

**Push**

**Introduction To Version Control**

* *A*version control*system is a special application that stores and manages every revision of your files and code.*
* *Many developers and organizations use version control to collaborate on source code, manage releases, and roll back to previous versions when bugs are discovered.*
* *When the developers makes the changes to the folder and save or commit them to host the server.*

Why we Use Version Control?

**Version control helps teams solve these kinds of problems**

* A version control system tracks every individual change made by each contributor and prevent concurrent work from conflicting.
* Changes made in one part of the software can be incompatible with those made by another developer working at the same time. This problem should be discovered and solved in an orderly manner without blocking the work of the rest of the team.
* Further, in all software development, any change can introduce new bugs on its own and new software can't be trusted until it's tested. So testing and development proceed together until a new version is ready.

**Types Of Version Control**

* There are three types of Version Control System
* File locks
* Centralised Version Control System
* Distributed Version Control System
* **FILE LOCK :** The other name of File Lock is Local Version Control System. In local Version Control System, no two developers can work together at same time.
* **Centralised Version Control System :** In this Version Control System, there is a single master copy of code base, and a pieces of code that typically worked on is locked. *so that only one developer is allowed to work on that part of the code at any one time*. *After checking code back in, the lock is released. So, it is available to others to checkout.*
* **Distrubuted Version Control System :** In this Version Control System,  *allow users to****work****independently in either a connected or disconnected environment.*

**What is Git ?**

* *D*istributed Version Control System keep tracks changes of files and directories present in the repository.
* *Allow users to****work****independently in either a connected or disconnected environment.*
* Stores information in data structure.

**Creating a empty Repository :**

* First you need to create a Directory(for example with name Repo)
* After creating, we need create a empty git repository in that Directory. For that we need to use following command
* **git init**
* If use that command it initializes with empty Repository init.

**File System of Git** **:**

* *To commit the file in a Git there are three stages*
* *UnTracked*
* *UnStaged*
* *Repository*
* **UnTracked files** *are the files created in git repository without commit and added to cache*
* **Unstaged files :**  *The UnTracked files added to cache. We can commit a file only when added to cache that is unstaged . This can be done by using following command*
* **git add .**
* *We can only commit an unstaged files. This can be done by using following command*
* *git commit –m “message”*

**Adding files to git:**

* *For adding u to need to follow below steps*
* *Check whether git is present in that directory or not*
* *If yes, check whether there is untracked are there or not. If yes, then add to cache and then commit by using following commands*
* *Git add .*
* *Git commit –m “message”*

**Status of a file or directory :**

* *In git, we check the status of a project whether it is untracked file or unstaged file. It can be done by using following command*
* *git status*

**Commit a file or directory :**

* *We can only commit an unstaged files. This can be done by using following command*
* *git commit –m “message”*

**Ignoring certain file system :**

* *In git, ignoring file system means ignore file to commit. This can be done by adding exclude file.*
* *In Ignoring certain file system, First we need to create exclude files .*
* *Global exclude files can be created using following command*
* *git config --global core.exclude ~/.gitingnore*
* *Now, we need create .gitignore file either in local remote area or global remote area*

**Git Logs :**

* *Git logs shows commit logs*
* *There are lot of git log commands*
* *git log –all displays all commit commands*
* *git log -3 displays n most recent commits*
* *git log –author “authorName” filter commits by author*
* *git log –commiter “name” filter commits by committer*
* *filtering commits by x days ago*
* *git log --before <date>*
* *git log –after <date*

**Cloning Repository :**

* *Git Clone* ***command downloads an existing Git repository to your local computer***
* *Git clone* [*https://github.com/sivakumarsagar/siva.git*](https://github.com/sivakumarsagar/siva.git)

**Branching :**

* *Allows to work more than one person.*
* *Usually a git is series/Collection of commits*
* *A branch in Git is simply a****lightweight movable****pointer to one of these commits. The default branch****name****in Git is master. As you initially make commits, you're given a master branch that points to the****last****commit you made*
* *You can see in which branch you are working by the command*
* *git branch*
* *You can also create a new branch by using command*
* *git branch <new branch name>*
* *You check in which branch you are working*
* *git checkout branchName*

**Git Merging :**

* *Merging two branches into target directory is called merging*
* *git merge <source branchName>*

**Git push :**

* ***git push****. The "****push****"****command****is used to publish new local commits on a remote server*
* *Before using "git push", make sure the correct local branch is checked out. Then, to perform the push, simply specify which remote branch you want to push to:*
* *git checkout <branchName> and then*
* *git push origin <branchName>*