**Version Control System -:** Also known as

Software Configuration Management

OR

Source Control Management (SCM)

**Need of Version Control System --?**

As developer , we write codes and keep them in files

One day what happen a client approached me to make

Their project .

Say the client project contains 100 files ,

When developer showcased the project , client replied that they want some changes (Day 1) .

Now to meet the requirement , Changes are made in some of the files .

Now after changes are done and presented to Client , client again asked for some changes (Day 2) .

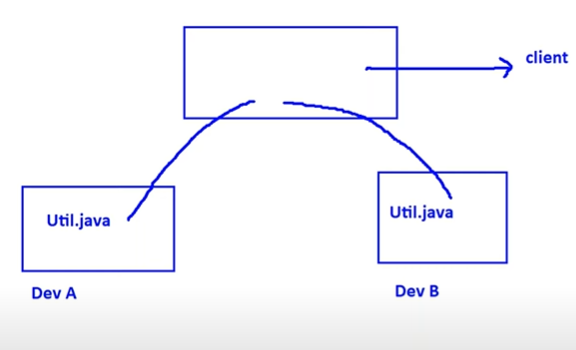
Again some changes made to that project .

Now when this time the project showcased to the client (day 3) , the client replied : Hey Dev I want to see the project as it was on Day 1 , can you show me ,,

At time developer felt helpless as he didn’t saved or manage code as it was on day 1st .

**That’s why we shouldn’t overwrite our code**

**Also in big organization , there are many developer working on the same project .**

* Its difficult to maintain different version of the project or task manually .
* Every changes should be tracked
  + When changes are done .
  + What changes are done
  + Who did the changes .
  + Changes should not be overwritten , If 2 developers are working on same project , their separate changes should be presented .
  + Parallel development and sharing code among different developer is also required.

The overall mechanism that help in such scenarios is called **Version Control**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Who use Version Control System and for what purpose**

Some common examples are -:

**Tester :** to manage the test scripts

**Architect :** To manage the design pattern in document forms

**Project Manager :** To manage the Excel sheets

And so on…..

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**How Version Control System Works**

**Working directory /Workspace :**  The folder where Developer keep their files , It has nothing to do with version control .

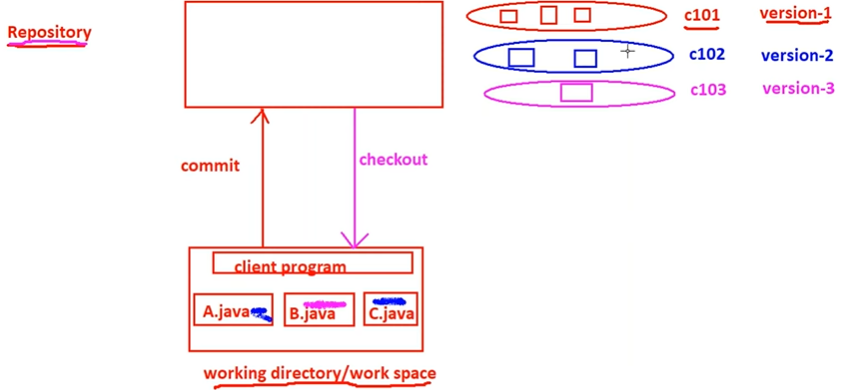
**Repository :** It is folder where we keep files along with their metadata . Here The version control is applicable .

**Client program:** The program responsible for transfer of file from working directory to repository and vice-versa

Example of client program is : git

**Commit :** The process of sending files from working directory to repository is called a commit

**commit id :**  Every commit has some unique id called “commit id” say c101 , this commit id contains information like who commit changes , by whom changes were made , when commit is done .

**Checkout :**  The process of transfer the files from Repository to Working Directory is called Checkout , sometimes referred as pull or clone .

Example : in working directory we have 3 files , then using some client program we commit the files to Repository , now a commit id generated with message as “version-1” ,

Developer makes some changes in 1st and 3rd file only , so when he commit this time a new commit id with message would be generated and 2nd file version remains the same .

Developer makes some changes to 2nd file now only changes are committed to 2nd file and not others and new version id with new message would be generated .

If developer want the file as 1st version ,he just need to checkout the 1st version from Repository to Working directory .

Benefits of Version Control System :

* We have different versions , and can choose any version as per our requirement .
* We can maintain metadata like
  + Commit message
  + Who done a commit
  + When a commit is done
  + What changes has been made
* Can share code to different Developers and collaborate
* Parallel development
* Also can provide the access control like
  + Who can read data .
  + Who can modify code

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

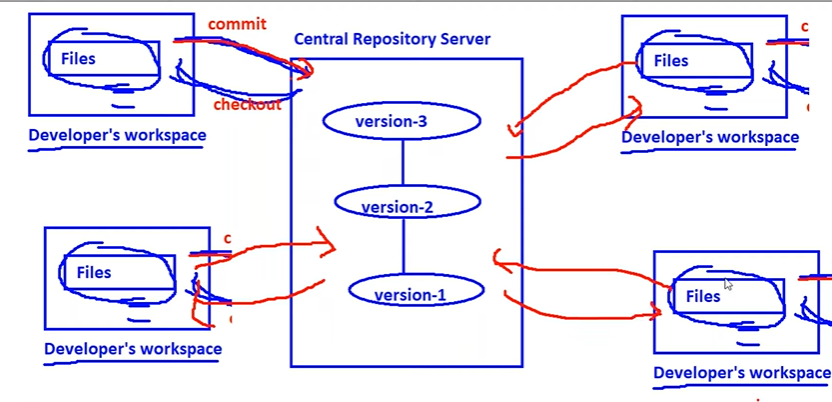
**Types of Version Control System :-**

* **Centralized Version Control System**
* **De-centralized /Distributed Version Control System**

**Centralized Version Control System -:**

**Centralized 🡪**  Contain only one central repository , developers need to connect with that central repository in order for version control .

It is very easy to setup and use

Example are : CVS , SVN , Perforce , ClearCase , TFS etc.

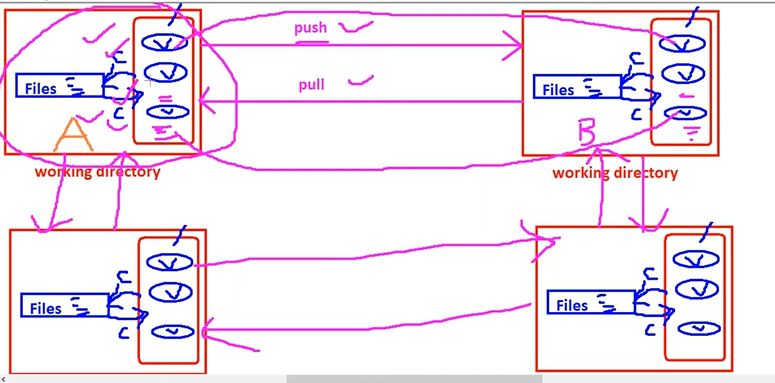
**Limitations:-**

* If something happened to the central repository server , the total codebase and files would be lost so there is a single point of failure . Recovery is difficult in this case .
* To perform checkout operation by connecting to central repository server same with the committing of files , if there is network outage , version control wouldn’t be available . So we have to make sure all the time that connection remains established .
* Performance would be affected as the operation we going to perform is not local operation , but remote operation (over the network) .
* If number of developer or files increases then centralized approach is not sufficient .

**Due to overcome these all drawbacks we goes for Distributed VCS**

**Distributed Version Control System -:**

In this type of VCS , Repository is distributed ,each developer have their own local repository , if there are say 4 developers then there will be 4 local repository each developer having one local repo.

****