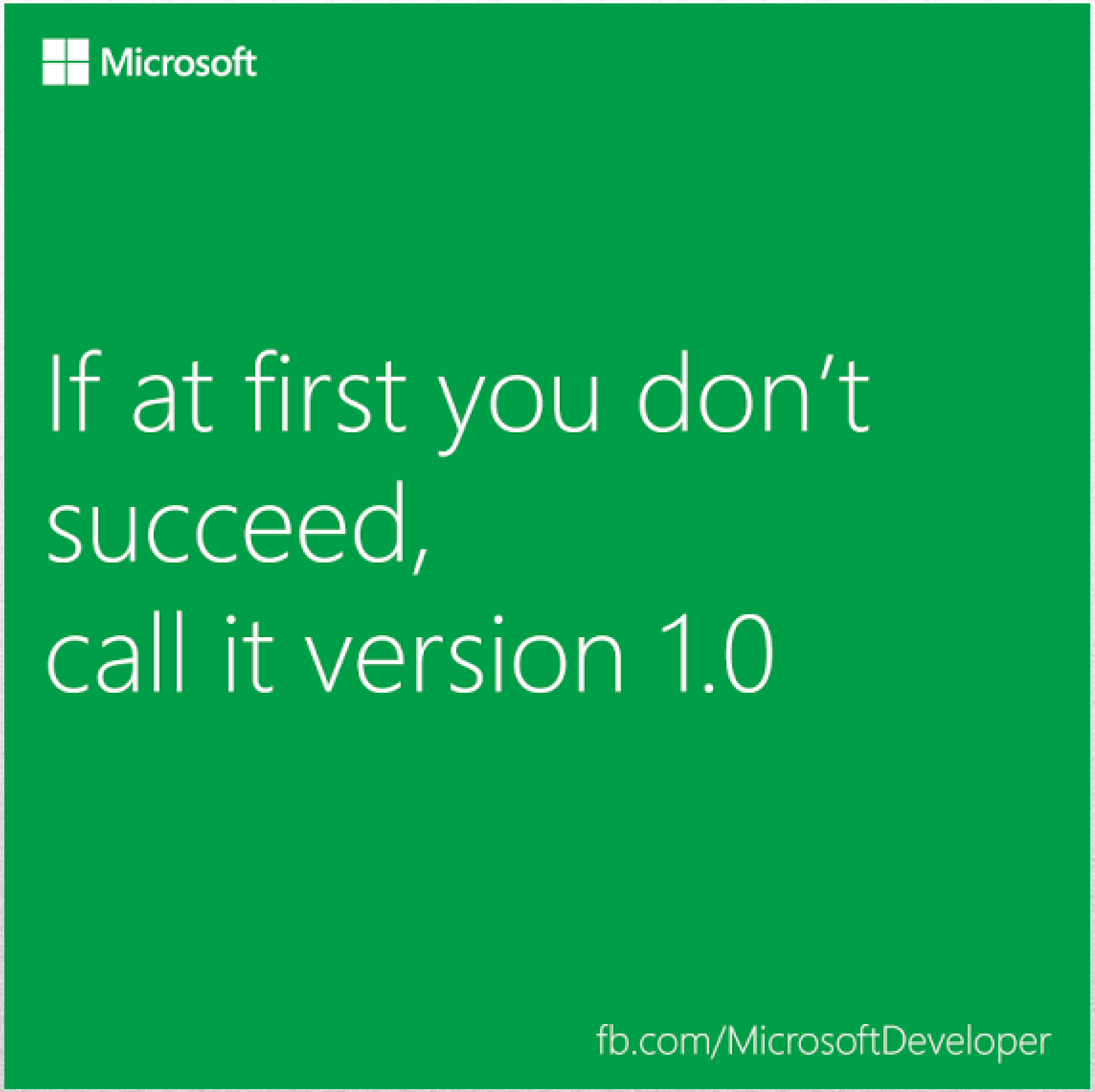
Introduction to Git

Version Control System

Abhishek Gupta

IIIT Jabalpur

14th Apr,2014


Microsoft

If at first you don't succeed,
call it version 1.0

fb.com/MicrosoftDeveloper

- What is Version Control System ?
- Where do you need one ?
- How did they evolve ?
- Why Git can be our best bet ?

Things you will learn !

- Travel back in time to correct mistakes
- Better handle situations where frequent changes happen on different parts of your digitalized files
- Options like undo & redo
- Maintaining multiple versions of the same file. (How ?)

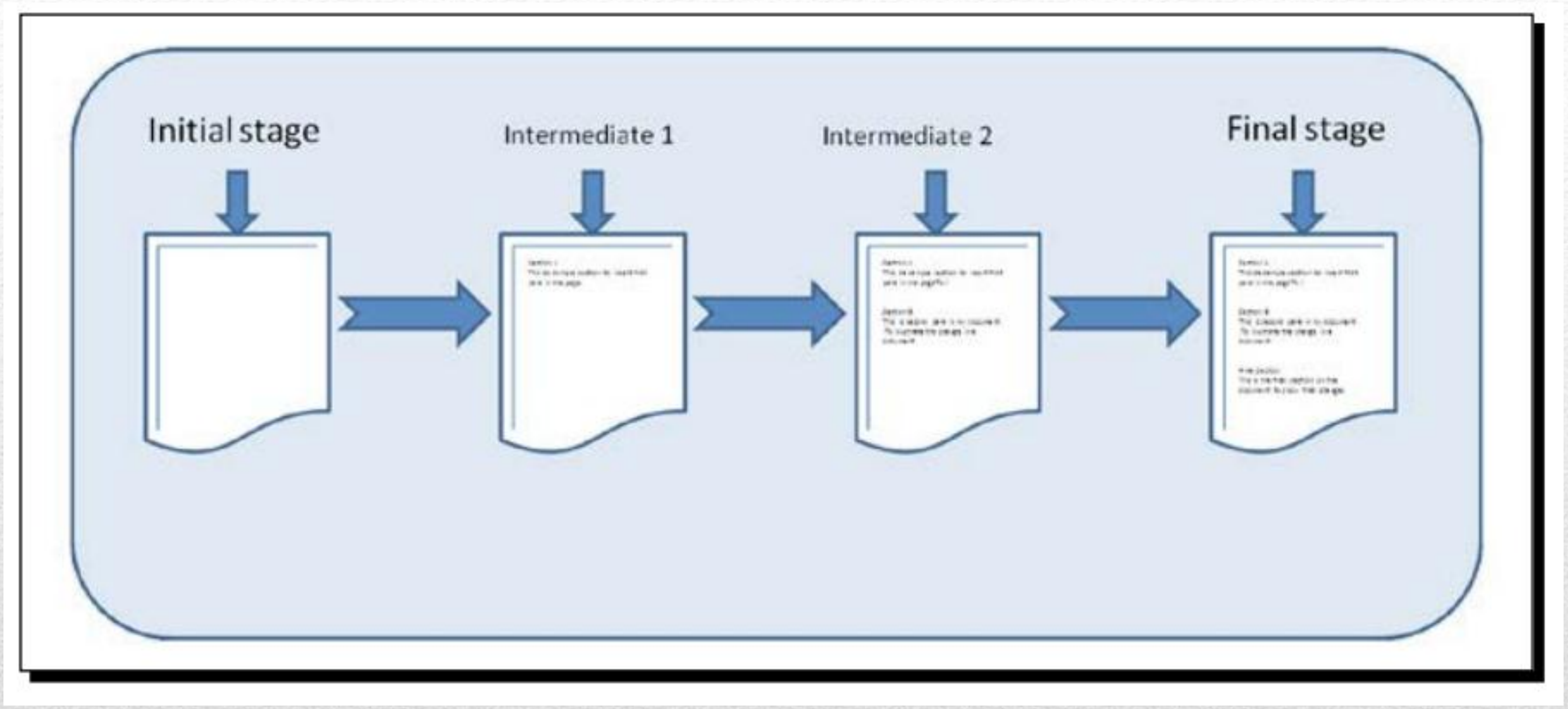
Any troubles ?

- A system capable of recording changes made to a file or a set of files over a period of time in such a way that it allows us to get back in time from the future to recall the specific version of that file
- A software package to allow you to monitor your files to tag the changes at different levels so that you can revisit those tagged stages whenever needed.
- Maintains history of changes made to your file.

What is VCS ?

- Can you maintain multiple versions of the same file under the same name, thus avoiding cluttering of files with small differences in their names mentioning their versions ?
- Do you have any means of marking a specific portion of your content in the file/files that you might need in future before changing them for present needs ?
- Are you satisfied with “backup” folders and updating them regularly ?
- Power to play with ‘flow of changes’ happening to your document.

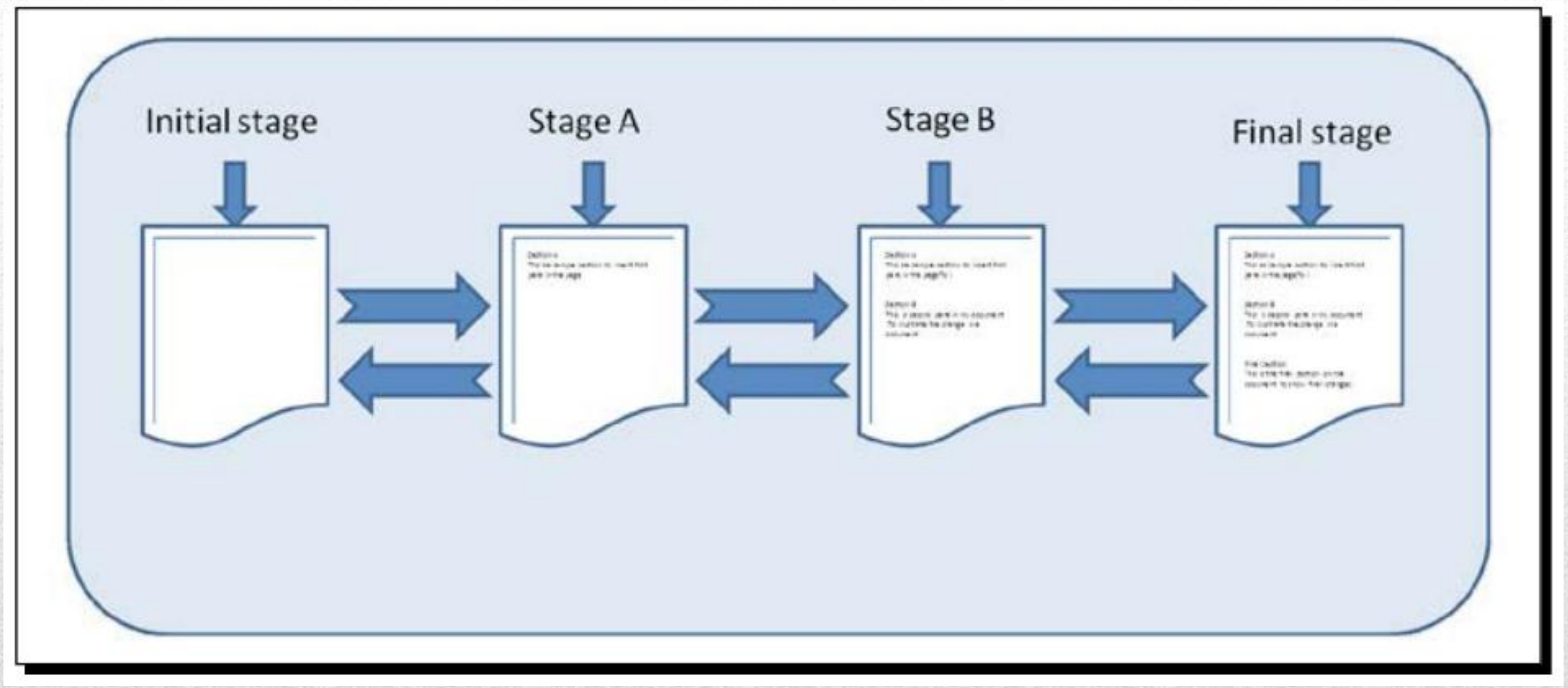
Why need a VCS ?



Why need a VCS ?

- Traditional – unidirectional (from left to right)
- "Save as" might help but not much of use !!
- VCS – multidirectional free flow context. You mark each and every change you consider important as a new stage and proceed with your content creation. This allows you to get back to any earlier stages that you have created, without data loss.
- You can literally jump to and fro between and across stages in any direction without any data loss.

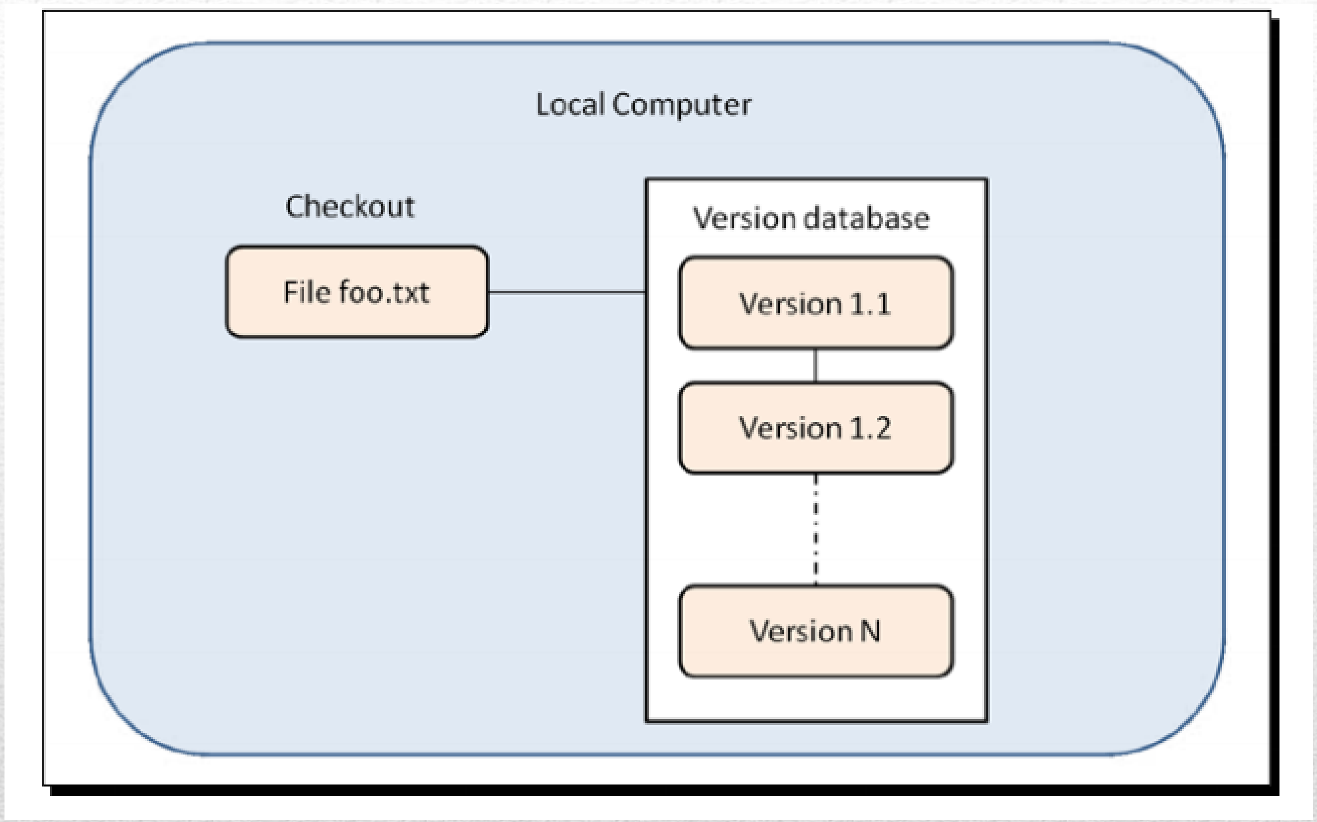
Why need a VCS ?



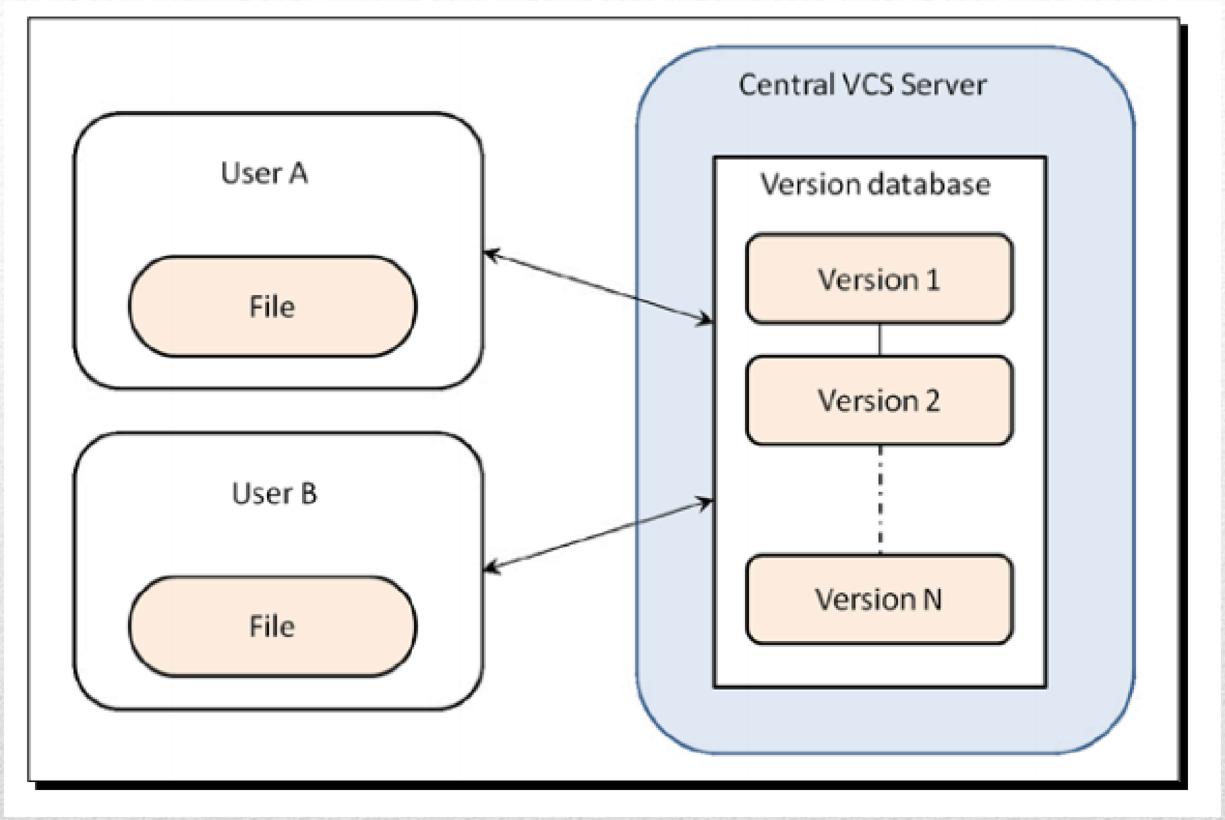
Why need a VCS ?

Types of VCS

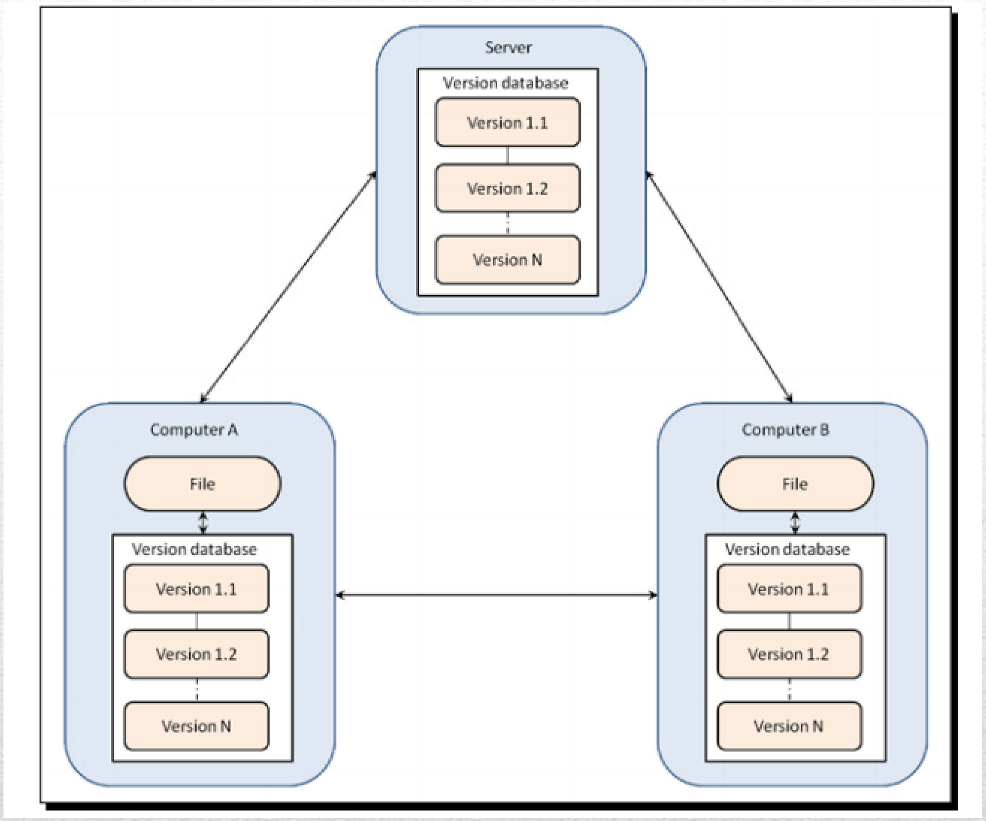
- Classification based on **mode of operation :**
  - Local VCS
  - Centralized VCS
  - Distributed VCS


Local Computer
Checkout
File foo.txt
Version database
Version 1.1
Version 1.2
Version N

Local VCS



Centralized VCS



Distributed VCS

Distributed VCS

- Has advantages of local VCS :
  - Making local changes without any concern of full time connectivity to the server
  - Not relying on single copy of files stored in the server.
- Has advantages of centralized VCS :
  - Reusability of work
  - Collaborative working, not relying on history stored on individual machines

Why Git ?

- Atomicity - no data loss or version mismatch due to partial operations
  - Banking
  - Seat reservation system
- Performance -Handles enormous units and process in seconds.
- Security – Git allows only when it knows!
  - SHA-1 hash used before storing.

- Now no talk ….. We shall take a walk ….
- Ready ?

Let’s dive in !

- Five important concepts :
  - Initiating the process
  - Adding your files to repository
  - Committing the added files
  - Checking out
  - Resetting

Suit up – getting ready for Git

- Its nothing but a process of pointing your finger towards that directory so that Git will know it has to monitor its contents for changes from them on.
- May use GUI/CLI.
- `git init /path/to/your/workbench/`
- Creates a .git directory inside workbench. It’s **hidden** and **read-only.** Guess what it contains?

Initiating

- Provide ways for Git to identify and group changes made to files present in any repository.
- Local configuration – particularly for current workbench
- Global configuration – applies to all repositories created using this installation.
- `git config --global user.name “Abhishek”`
- `git config --local user.name “Abhishek”`
- `git config --local user.email “Abhishek”`
- `git config -l`

Initiating – Configure Git

- From hereon anything we ask Git to monitor shall be called as ‘repository’.
- As simple as copy and paste
- Create a file and then add it to repository.
- `git status`
- `git add “abc.txt”`

Adding files to directory

- Avoid certain files into one’s repository.
- `git add .` is equivalent to *Ctrl +i* in GUI mode.
- But all files get added without exception ☹
- .gitignore to rescue. Inside the repository.
- Enter the names or pattern of filenames for Git to exclude them. ☺
- You can also move a file from staged area to unstaged area :: `git reset filename.extension`

Ignore ‘em

- Still, files are not under ‘Version Control’
- Associate informative comment with each commit.
- `git commit -m “your message”`.
- After commit, Git provides a status commit ID.
- From hereon, any changes made to the file will be relative.

Committing the added files

- It is the process that helps you jump to and fro between the changes that you have made in any single file or entire subset of files in your repository at the time of commitment .
- Gitk gives you a graphical view to perform various kinds of operations such as visualizing the repository, tagging, resetting, and so on.
- SHA1 ID is the commit ID. It helps us to travel back in time. Branch->Checkout.
- Revert to your latest changes by selecting Branch->Checkout-> Local branch.

Checking out

- `git log` – shows history of repository
- `git checkout <ID>` - To travel back in time
- `git checkout master` – To have latest changes

Checking out

Cloning

- Cloning- exact replica of working repository **along with history**.
- The `git clone` command copies an existing Git repository.
- Has its own history, manages its own file and is a completely isolated environment from the original repository.
- Enables repo- to – repo collaborations.
- `git clone ssh://abhishek@think-pots.appspot.com/mindTheGap/mtg.git`
- `cd mtg`
- `mtg` folder is initialized automatically.

- `git clone https://github.com/yourUserName/iiit.git`
- `git remote add upstream https://github.com/abhi10005/iiit.git`
- `git fetch upstream`
- `git merge upstream/master`
- `git push origin master`

$\left.\begin{array}{l}\texttt{git fetch upstream}\\ \texttt{git merge upstream/master}\end{array}\right\}$ These steps required to update changes made to your repository by someone else.

Github hands-on

- Offline help available
  - `git help`
  - `git help operation_keyword` gives complete reference sheet of that particular operation.

Help Me!

- Splitting the load – Distributed working with Git
- Controlling Git functions
- Git on text based files
- Branching with Git
- www.atlassian.com/git/tutorial

Further ahead

- GROW to LEARN and LEARN to GROW !!

Thank You !!!