Version Control - GIT

Avinash Patel

Agenda

Introduction

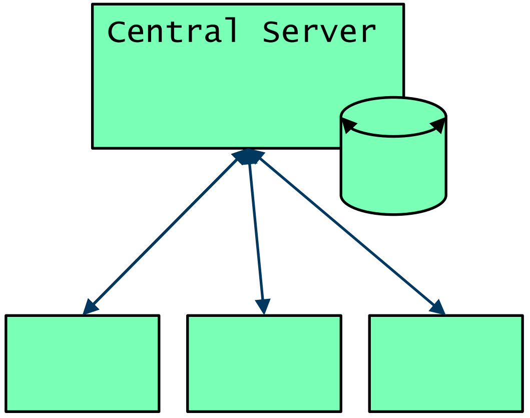
- In SDLC process, non-linear workflows and distributed framework are imminent.
- It is common that a piece of code is being accessed and possibly edited by a geographically dispersed team.
- Maintaining Data Integrity is very crucial when many team members (Developer/Tester) work on same files.
- Revision control is an efficient way to address the problem of sharing files.
  - It is also called as Version Control and Source Control.
  - Each of these revisions is typically identified by Time Stamps

What is version control ?

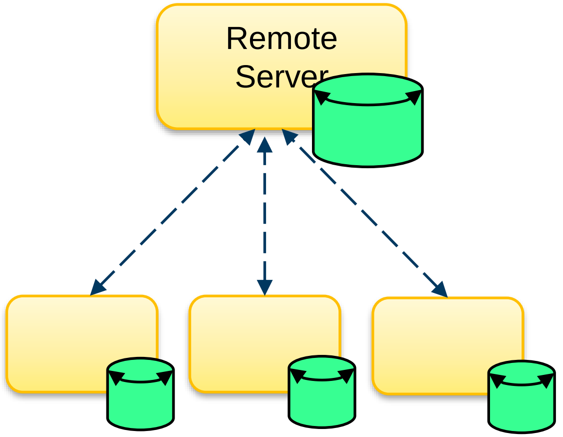
- Version control is a system that records changes to a file or set of files over time. It helps to recall or recover specific version as and when required
  - i.e. a system which allows the management of a code base

- Version control enables to roll back to the previous state of a file or files or a entire project.
  - It allows multiple versions to exist simultaneously
  - It compare changes over time
  - It checks the last modification
  - It easily tracks and recover with very little overhead

Types of Version Control (VC)



Centralized VC

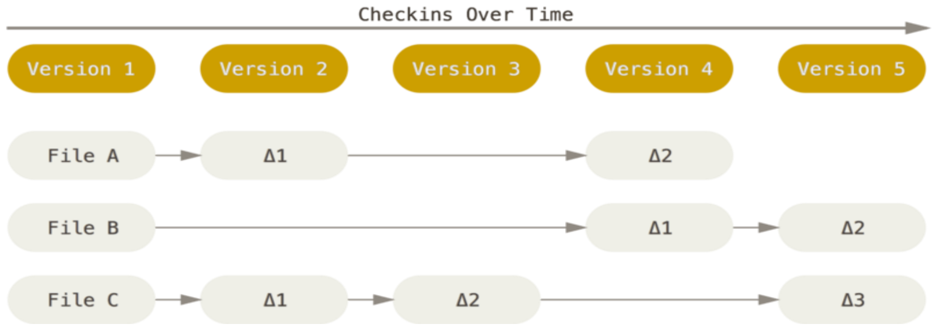


Distributed VC

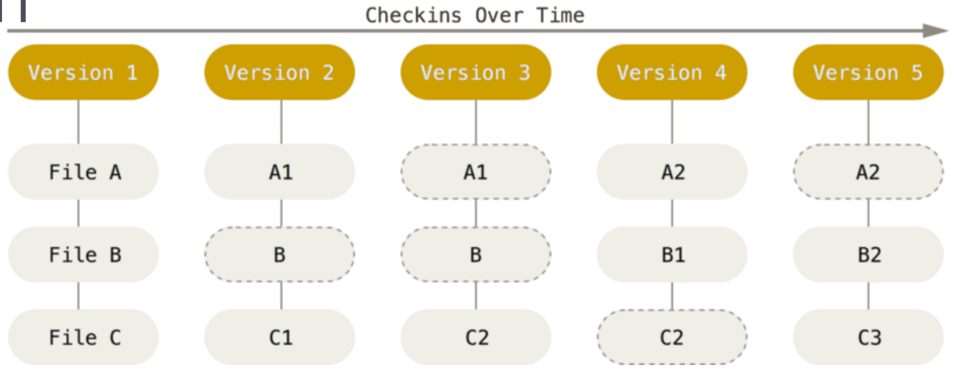
Introduction to GIT

Introduction

- Most Version Control systems



- GIT



Features of GIT

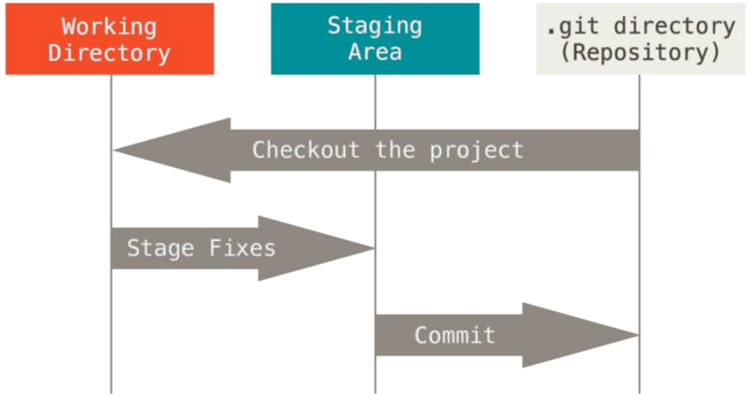
- Almost Every Operation Is Local:
  - Most operations in Git need only local files and resources to operate. No other information is needed from another computer on a network

- Git Has Integrity
  - Everything in Git is check-summed before it is stored and is referred to, by that checksum.
  - It’s impossible to change the contents of any file or directory without Git knowing about it

- Git Generally Only Adds Data
  - When actions are done in Git in the form of commands, nearly all of them only add data to a Git repository

GIT Basics- Three states

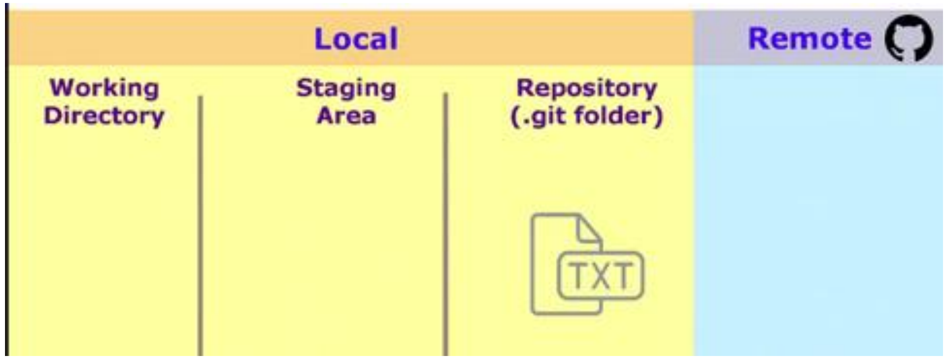
- Three stages are
  - **Committed**: The data is safely stored in local database.
  - **Modified**: It implies that the file is changed and yet to be committed into database.
  - **Staged**: It means that modified file is marked in its current version to go into next commit snapshot.



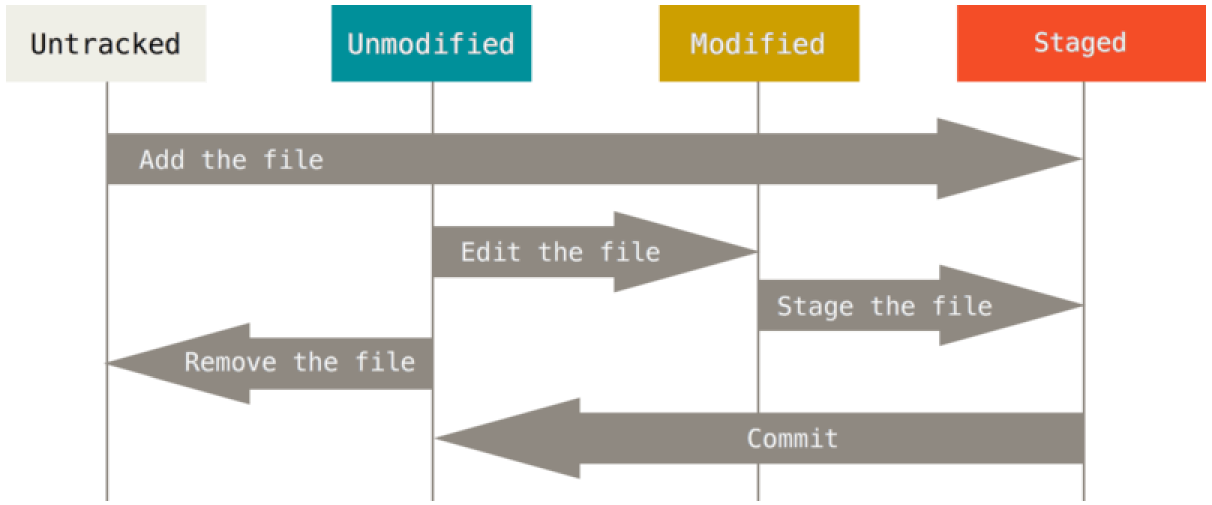
- Three main sections of a GIT project:
  - Working directory
  - Staging area
  - GIT directory

Git workflow

- Data can be placed in central repository (remote) from local git repository



The lifecycle of the status of files



Getting Started

- To download git for windows

  https://git-scm.com/download/win

- Signup and/or Create your own remote (public/private) repository http://wosggitlab.wipro.com

Basic Set UP / Getting Started with GIT

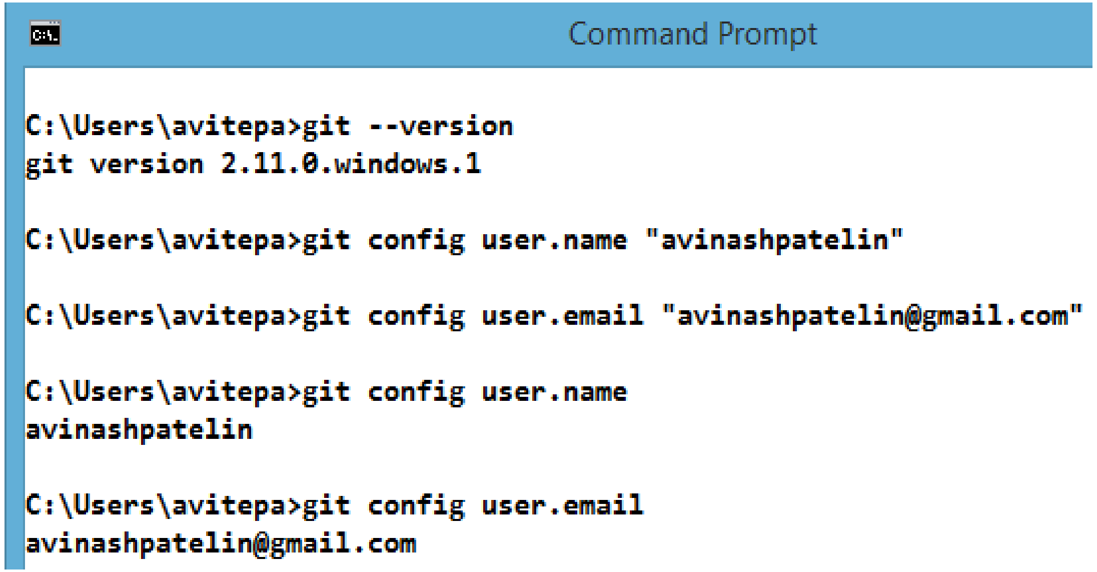
- Check GIT version
  - ➢ git --version

    git version 2.11.0.windows.1
- Create a distributed public repository at http://wosggitlab.wipro.com
  - Note: For example create “MyRepo.git” as a public repository
- Define user account's default identity
  - ➢ git config --global user.name “Any Valid User Name"
  - ➢ git config --global user.email your_email@whatever.com
  - ➢ git config --global core.editor “’C:/Program Files(x86)/Notepad++/notepad++.exe’ -multiInst -nosession"

Sensitivity: Internal & Restricted

Checking the Settings

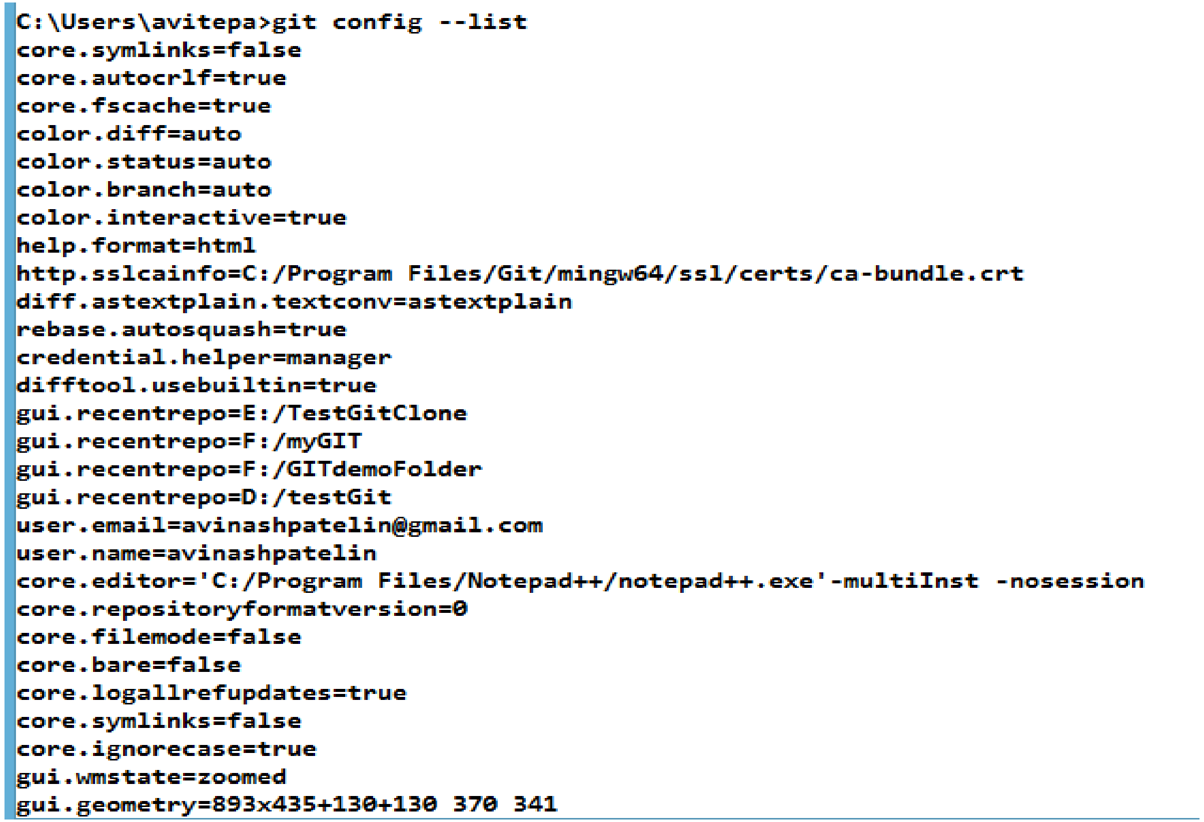
- For checking individual values of the keys
  - git config user.name
- To view all settings use the below command
  - git config –list



Command Prompt

```
C:\Users\avitepa>git --version
git version 2.11.0.windows.1

C:\Users\avitepa>git config user.name "avinashpatelin"

C:\Users\avitepa>git config user.email "avinashpatelin@gmail.com"

C:\Users\avitepa>git config user.name
avinashpatelin

C:\Users\avitepa>git config user.email
avinashpatelin@gmail.com
```

Checking the Settings contd..



```
C:\Users\avitepa>git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
credential.helper=manager
difftool.usebuiltin=true
gui.recentrepo=E:/TestGitClone
gui.recentrepo=F:/myGIT
gui.recentrepo=F:/GITdemoFolder
gui.recentrepo=D:/testGit
user.email=avinashpatelin@gmail.com
user.name=avinashpatelin
core.editor='C:/Program Files/Notepad++/notepad++.exe'-multiInst -nosession
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
gui.wmstate=zoomed
gui.geometry=893x435+130+130 370 341
```

Basic Commands

Working with Git Repository

- Option 1: Place an existing project or directory into Git (Remote repository)

- Option 2: Have a cloned copy of an existing Git (Remote repository)

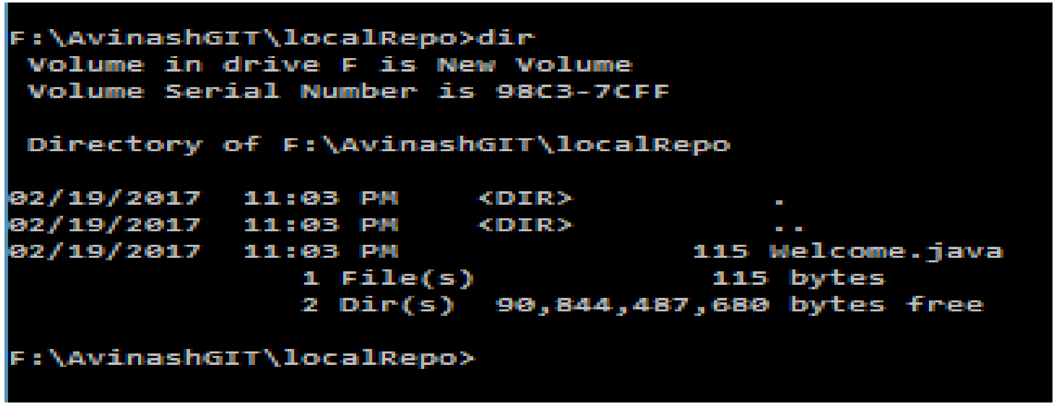
Option1: Create local project

- Create a folder ‘localRepo’ and initialize as local repository
  - **F:\AvinashGIT\localRepo>git init**

```
Initialized empty Gitrepository in F:/AvinashGIT/localRepo/.git/
```

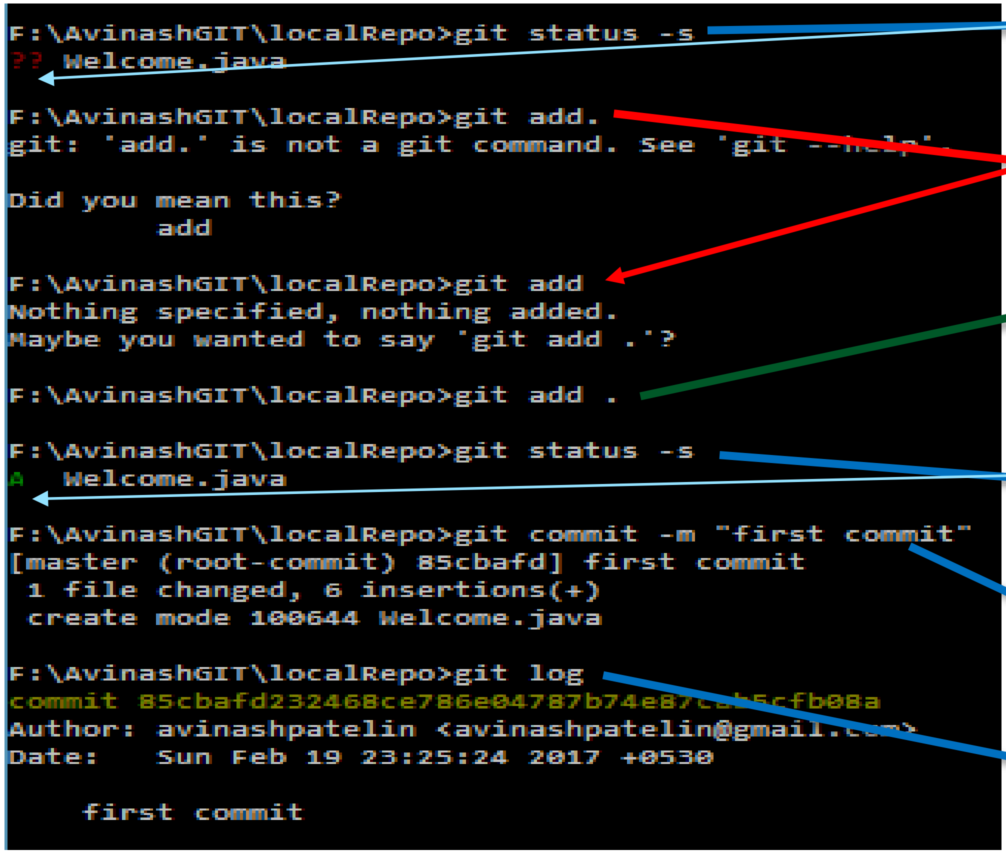
Note: It creates a new hidden subfolder named .git. This folder is used to contain all of your necessary repository files. At this point, nothing in that project is tracked.

- Create a file “Welcome.java” within ‘localRepo’ folder



```
F:\AvinashGIT\localRepo>dir
 Volume in drive F is New Volume
 Volume Serial Number is 98C3-7CFF

 Directory of F:\AvinashGIT\localRepo

02/19/2017  11:03 PM    <DIR>          .
02/19/2017  11:03 PM    <DIR>          ..
02/19/2017  11:03 PM               115 Welcome.java
               1 File(s)            115 bytes
               2 Dir(s)  90,844,487,680 bytes free

F:\AvinashGIT\localRepo>
```

Option1: Steps to Commit to Local Repo.



```
F:\AvinashGIT\localRepo>git status -s
?? Welcome.java

F:\AvinashGIT\localRepo>git add.
git: 'add.' is not a git command. See 'git --help'.

Did you mean this?
        add

F:\AvinashGIT\localRepo>git add
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'?

F:\AvinashGIT\localRepo>git add .

F:\AvinashGIT\localRepo>git status -s
A  Welcome.java

F:\AvinashGIT\localRepo>git commit -m "first commit"
[master (root-commit) 85cbafd] first commit
 1 file changed, 6 insertions(+)
 create mode 100644 Welcome.java

F:\AvinashGIT\localRepo>git log
commit 85cbafd232468ce786e04787b74e87cab5cfb08a
Author: avinashpatelin <avinashpatelin@gmail.com>
Date:   Sun Feb 19 23:25:24 2017 +0530

    first commit
```

1.Status after init

2. add a specific file or everything

3. Correct One:
add .

4. Status after add

5. Commit the file

6. Check the log

Sensitivity: Internal & Restricted

Option1: push local copy to remote repository (gitlab)

- Step1: git config remote.origin.url <url>

  Example: git config remote.origin.url http://wosggitlab.wipro.com/AVITEPA/demo.git

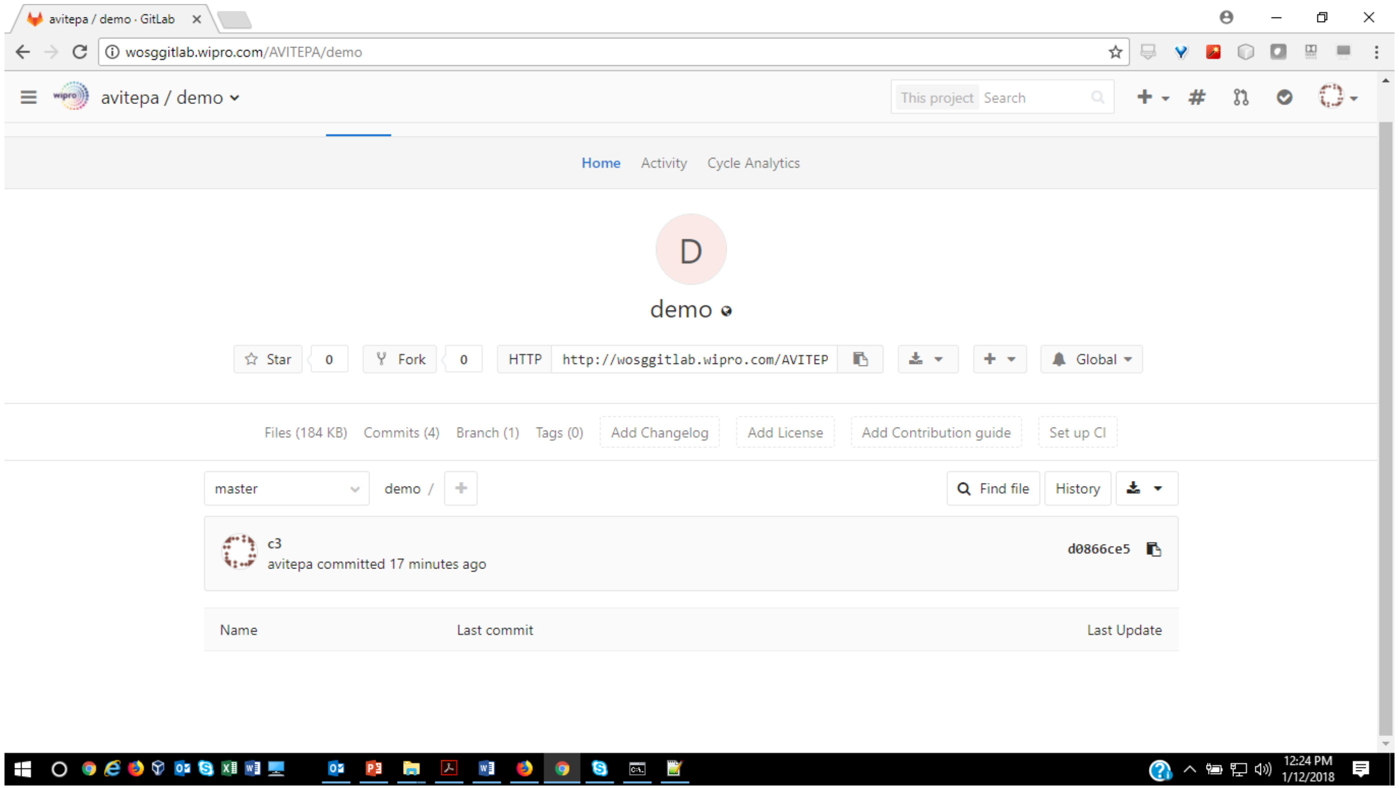
  Note: This project should have been created in gitlab already

- Step2: Push files from local to remote repository
  - **F:\AvinashGIT\localRepo>git push -u origin master**

Sensitivity: Internal & Restricted

Option1: Verify Central repository

- @ http://wosggitlab.wipro.com/AVITEPA/demo



avitepa / demo · GitLab

wosggitlab.wipro.com/AVITEPA/demo

avitepa / demo

Home Activity Cycle Analytics

demo

Star 0 Fork 0 HTTP http://wosggitlab.wipro.com/AVITEP Global

Files (184 KB) Commits (4) Branch (1) Tags (0) Add Changelog Add License Add Contribution guide Set up CI

master demo / Find file History

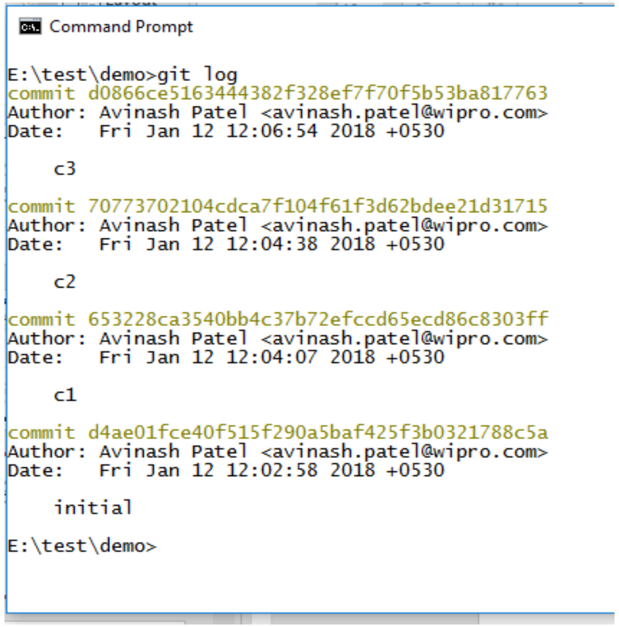
c3
avitepa committed 17 minutes ago d0866ce5

Name	Last commit	Last Update

12:24 PM 1/12/2018

Option 2

- Get a copy of an existing Git repository from remote repository
  - **F:\git clone http://wosggitlab.wipro.com/AVITEPA/demo.git**
- Use `git log` and observe the previous version track details



```
E:\test\demo>git log
commit d0866ce5163444382f328ef7f70f5b53ba817763
Author: Avinash Patel <avinash.patel@wipro.com>
Date:   Fri Jan 12 12:06:54 2018 +0530

    c3

commit 70773702104cdca7f104f61f3d62bdee21d31715
Author: Avinash Patel <avinash.patel@wipro.com>
Date:   Fri Jan 12 12:04:38 2018 +0530

    c2

commit 653228ca3540bb4c37b72efccd65ecd86c8303ff
Author: Avinash Patel <avinash.patel@wipro.com>
Date:   Fri Jan 12 12:04:07 2018 +0530

    c1

commit d4ae01fce40f515f290a5baf425f3b0321788c5a
Author: Avinash Patel <avinash.patel@wipro.com>
Date:   Fri Jan 12 12:02:58 2018 +0530

    initial

E:\test\demo>
```

Points to ponder

- *config* (*global* & *list*)
    - user.name <optional>
    - user.email <optional>
    - core.editor <optional>
    - remote.origin.url <optional>
- *init*
- add (. Or <filename>
- status (-s)
- *log*
- *commit* (-m <name>)
- Local repository to a remote repository - config to push
- *push* - Actual push
- *clone <url>*
- -- version

Branching & Merging

Sensitivity: Internal & Restricted

Branch

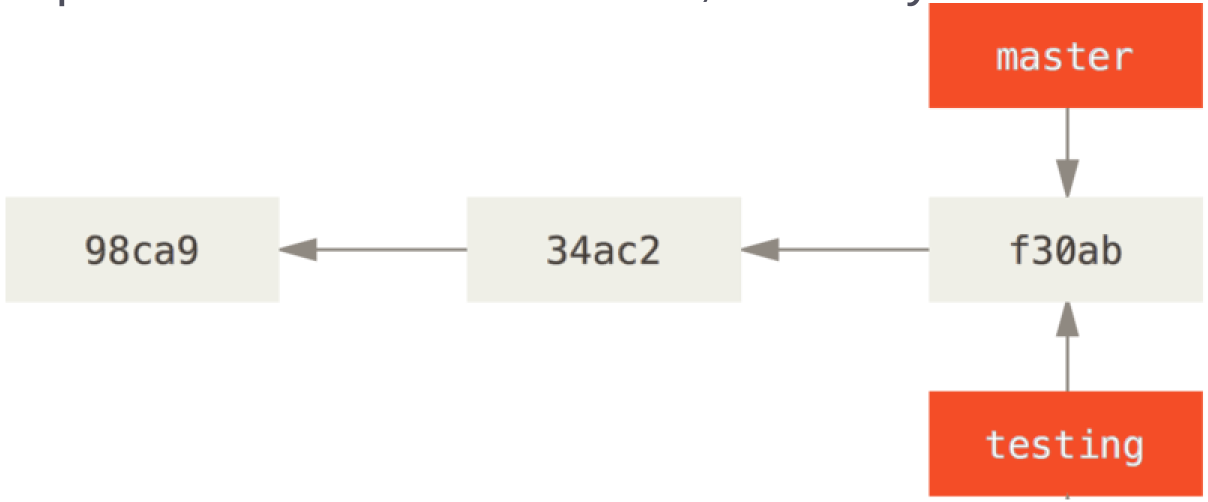
- A branch in Git is simply a movable pointer to one of the commits.

- In Git, the default branch name is ‘master’.
  - The master branch will point to the latest commit, as the commits are made
  - “master” branch is not a special branch When ‘git init’ command gets executed, It gets created by default.

Create a New Branch

- When a new branch is created, a pointer also gets created to move around.
- Use the command ‘git branch <branch name>’ to create a new branch

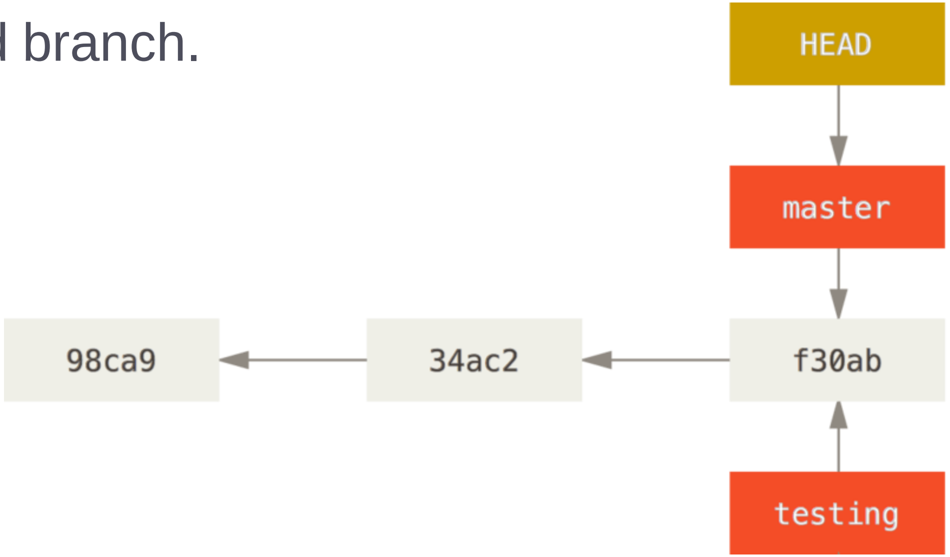
For example: ‘git branch testing’ creates a new branch called ‘testing’. It also creates a new pointer to the same commit, currently on.



HEAD pointing to a branch

- Git uses a special pointer called HEAD to know the current branch.
  - This pointer pointing to the local branch. In the diagram given below, it points to master

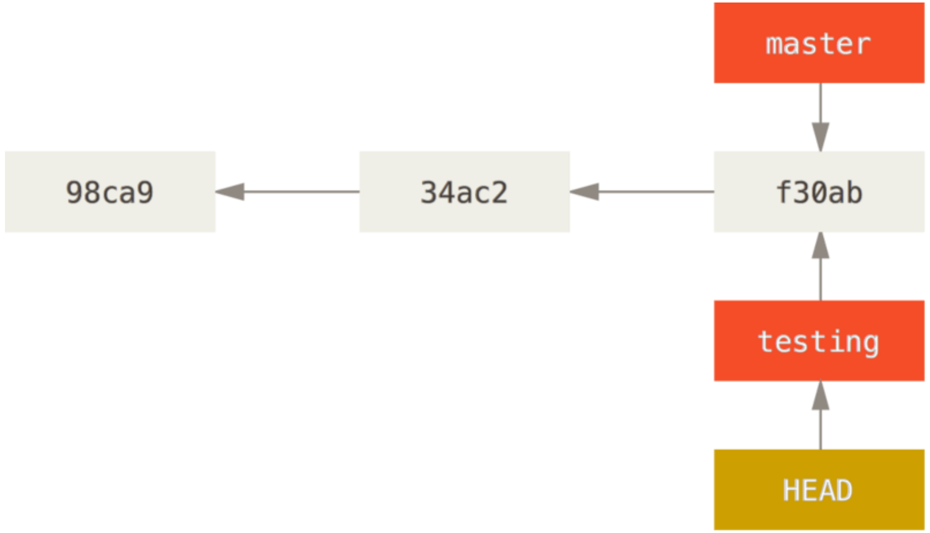
- 'git branch' command creates a new branch. However it doesn't switch to the newly created branch.



Switch to a Branch

- To switch to an existing branch, execute the command:
  - git checkout <name of the branch>

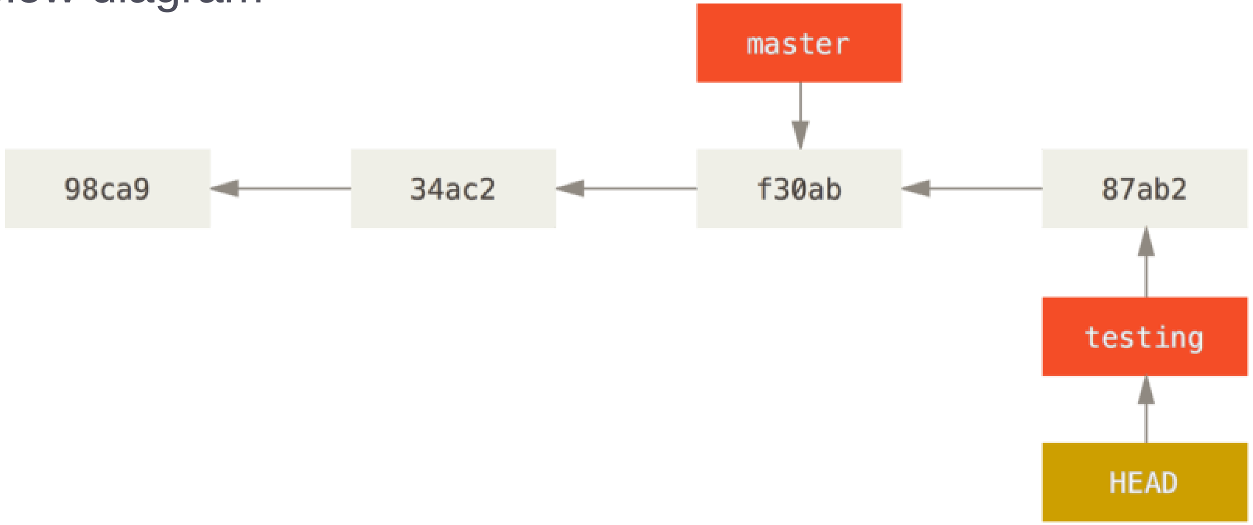
    For example : git checkout **testing**
  - This command makes the HEAD, to point to the **testing** branch.



Commit at the branch level

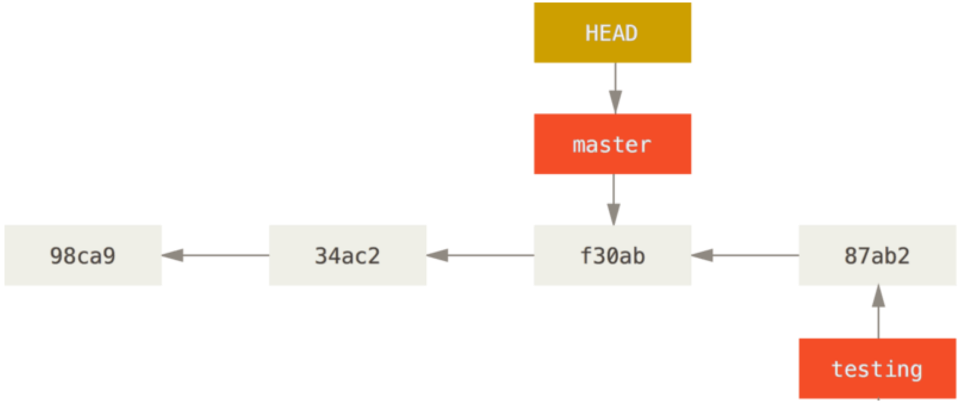
- Make few changes and commit at the branch level
  - git commit -a -m ‘few changes have been made‘

- As a result, ‘Testing’ branch pointer has moved forward along with HEAD. However master branch pointer still points to earlier commit as shown in the below diagram



Merge

- Switch back to master branch by using the command:
  - git checkout master



  - This command makes the HEAD pointer to point to master branch again. Thereafter, any change made will reflect on master only

- Use the below command to merge branch (testing) with master
  - git merge <name of the branch>

Thank You