TPortfolio 5

What is Version Control?

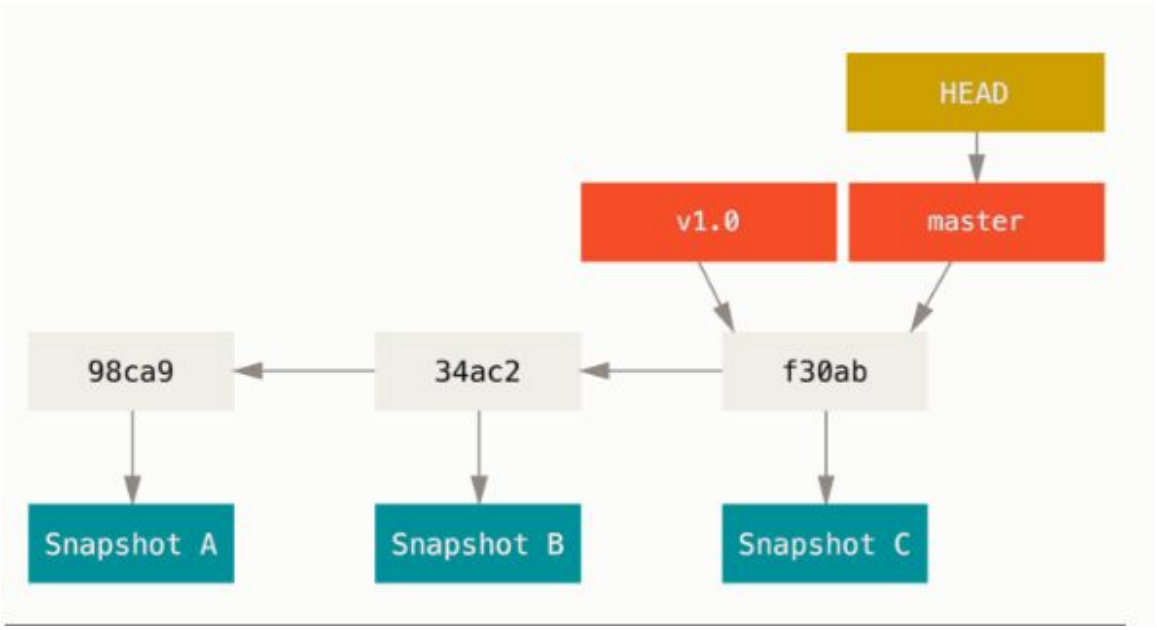
Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Why use Version Control?

- It provides **one method** for an entire team to use; everybody operates under the same 'ground rules'.
-
- Changes are **orderly vs. chaotic**, saving development time
-
- The ability to track changes promotes **accountability** and makes it easier to find the right persom to **solve problems** in the materials maintained.
-
- A **list of exact changes made can be generated quickly and easily**, making it easier to advise users of the information on how it has changed from version to version.
-
- It is **easy to 'roll back' to an earlier version** of the information, if a serious mistake was made during a change.

Branching

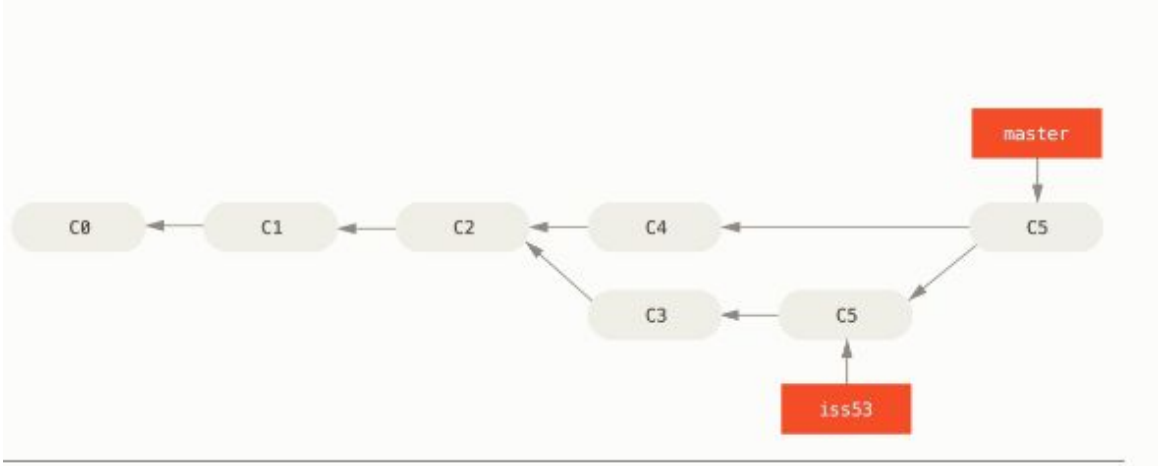
A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, it moves forward automatically.



A branch and its commit history

Merging

Merging (also called integration) in version control, is a fundamental operation that reconciles multiple changes made to a version-controlled collection of files. Most often, it is necessary when a file is modified by two people on two different computers at the same time. When two branches are merged, the result is a single collection of files that contains both sets of changes.



A merge commit

Why is merging necessary

Basically it is used to prevent the complete history of the project and prevent the risk of re writing public commit.

Merging Conflicts

If two people changed the same lines in that same file, or if one person decided to delete it while the other person decided to modify it, Git simply cannot know what is correct. Git will then mark the file as having a conflict - which you'll have to solve before you can continue your work. We can use the following command to solve the conflict.

$git status

Now we can see which file is causing the conflict. We can use some tools e.g mergetool created by git which handle the merging conflicts.The user will be able to choose which file to keep and which to dispose.We can use the following command for that

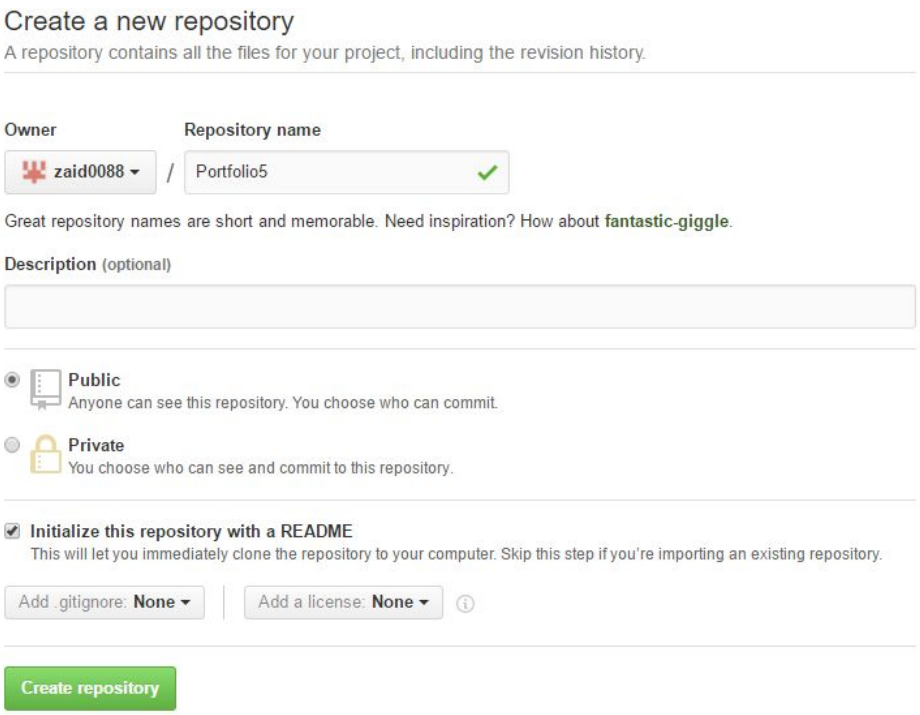
$git mergetool

Check In and Check out

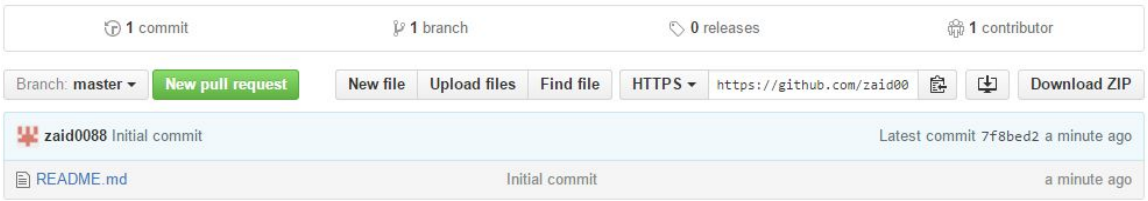
Check in is the process when you upload the file to the repository after some changes have been done to the file. So the new file will get a new serial number so that other people will recognise it.On the other hand check out is to download or clone a repository from a host.

Demonstration

First we need to create a new repository.



After the repository have been created the user need to copy thr repository url and clone it to create a local repository.



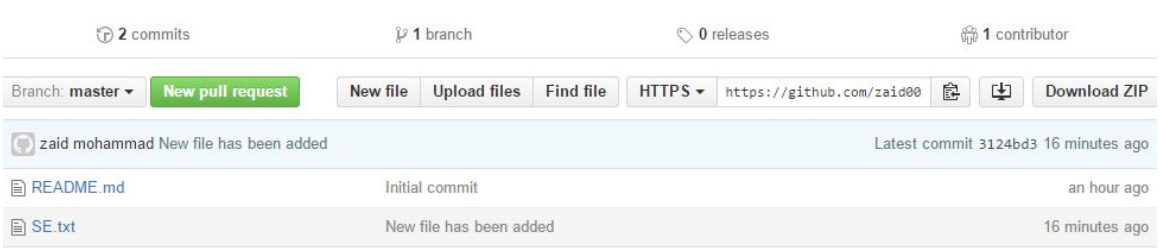
The user have to lauch gitshell and type in the command git clone followed by the copied link. This will clone the repository. After that the user need to change the directory.

```
:\Users\Zaid-PC\Documents\GitHub> ls
:\Users\Zaid-PC\Documents\GitHub> git clone https://github.com/zaid0088/Portfolio5.git
loning into 'Portfolio5'...
emote: Counting objects: 3, done.
emote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
npacking objects: 100% (3/3), done.
hecking connectivity... done.
:\Users\Zaid-PC\Documents\GitHub> ls


    Directory: C:\Users\Zaid-PC\Documents\GitHub


ode                LastWriteTime         Length Name
---                -------------         ------ ----
d-----        10-May-16   5:31 PM                Portfolio5


:\Users\Zaid-PC\Documents\GitHub> cd Portfolio5
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡]> ls


    Directory: C:\Users\Zaid-PC\Documents\GitHub\Portfolio5


ode                LastWriteTime         Length Name
---                -------------         ------ ----
-a----        10-May-16   5:31 PM             12 README.md
```

After that the user can add or update the file into the repository, the user can add a commiting message, -m tag is used for that. The file will not be updated yet. The user has to use the push command for that.

```
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡]> git add "SE.txt"
atal: pathspec 'SE.txt' did not match any files
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡]> git add "SE.txt"
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡ +1 ~0 -0 ~]> git commit -m "New file has been added"
master 3124bd3] New file has been added
Committer: zaid mohammad <zaid mohammad>
our name and email address were configured automatically based
n your username and hostname. Please check that they are accurate.
ou can suppress this message by setting them explicitly:

   git config --global user.name "Your Name"
   git config --global user.email you@example.com
```

```
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ↑ +0 ~1 -0 !]> git push
sernamе for 'https://github.com': zaid0088
assword for 'https://zaid0088@github.com':
ounting objects: 3, done.
elta compression using up to 4 threads.
ompressing objects: 100% (2/2), done.
riting objects: 100% (3/3), 270 bytes | 0 bytes/s, done.
otal 3 (delta 0), reused 0 (delta 0)
o https://github.com/zaid0088/Portfolio5.git
   7f8bed2..3124bd3  master -> master
```



Now we can see the file in our repository.

If the user has some collaborators he can use the pull command to to get the newly added file from remote repository to local repository.

```
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡ +0 ~1 -0 !]> git pull
```

.git	10-May-16 5:54 PM	File folder	
collaborator.txt	10-May-16 5:59 PM	Text Document	0 KB
README.md	10-May-16 5:31 PM	MD File	1 KB
SE.txt	10-May-16 5:52 PM	Text Document	1 KB

User can create a branch if he wants to make changes without affecting the master repository.

```
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡ +0 ~1 -0 !]> git branch
master
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡ +0 ~1 -0 !]> git checkout -b nb
      SE.txt
itched to a new branch 'nb'
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb +0 ~1 -0 !]> git push --set
tal: The current branch nb has no upstream branch.
 push the current branch and set the remote as upstream, use

  git push --set-upstream origin nb

\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb +0 ~1 -0 !]>  git push --set-upstream origin nb
tal 0 (delta 0), reused 0 (delta 0)
 https://github.com/zaid0088/Portfolio5.git
 [new branch]      nb -> nb
anch nb set up to track remote branch nb from origin.
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb ≡ +0 ~1 -0 !]>
```

```
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb ≡ +0 ~1 -0 !]> ls


    Directory: C:\Users\Zaid-PC\Documents\GitHub\Portfolio5


lode                LastWriteTime         Length Name
---                 -------------         ------ ----
-a----        10-May-16   5:31 PM             12 README.md
-a----        10-May-16   5:52 PM             19 SE.txt


:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb ≡ +0 ~1 -0 !]> git add "bt.txt"
atal: pathspec 'bt.txt' did not match any files
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb ≡ +0 ~1 -0 !]> git add "bt.txt"
:\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb ≡ +1 ~0 -0 | +0 ~1 -0 !]>
```

File added to new repository

```
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [nb ≡ +1 ~0 -0 | +0 ~1 -0 !]> git checkout master
      SE.txt
      bt.txt
itched to branch 'master'
ur branch is up-to-date with 'origin/master'.
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡ +1 ~0 -0 | +0 ~1 -0 !]> git merge nb
ready up-to-date.
\Users\Zaid-PC\Documents\GitHub\Portfolio5 [master ≡ +1 ~0 -0 | +0 ~1 -0 !]> git push
```

Switched back to master and after that merge and pushed to the repository