

Portfolio 5

What is version control? Version control is a system that records all the changes of a files over time. The users who have permission to access the files can recall the specific versions of the file.

Version control is important because it record the detailed history of all the changes made to the file and who made them. Without this history, it would be almost impossible to have many contributors working simultaneously on the same project because if anything were to go wrong, it will be difficult to locate the bugs from millions line of code. But with the use of version control system, it allows the project managers to locate bugs easily and able to revert the code back to a previous working version.

Git is one of the many Version Control solution available now. The system provides facilities such as check-out.

Git HEAD

Git Head is a reference to the currently checked out commit. But in normal states, it is a symbolic reference to the branch checked out. The branch itself is a reference to the commit at the tip of branch.

Git check-out

Git checkout is used to switch branches or restore working tree files. It updates files in the working tree to match the version in the index. Git checkout will update HEAD to set the specified branch as current branch if no path are provided. Checking out a commit will cause the entire working directory match to the commit. This can be used to view an old version of the file without altering the current state of file. Checking out a file allow user to view the old version of file without touching the working directory.

To return to the master branch or return to the current state of project, use `git checkout master.`

To check out a previous version of a file, use `git checkout <commit> <file>.` This will turn the file in the working directory into an exact copy of the file from commit and add to the staging area.

To update all the files in the working directory to match the commit, use `git checkout <commit>.`

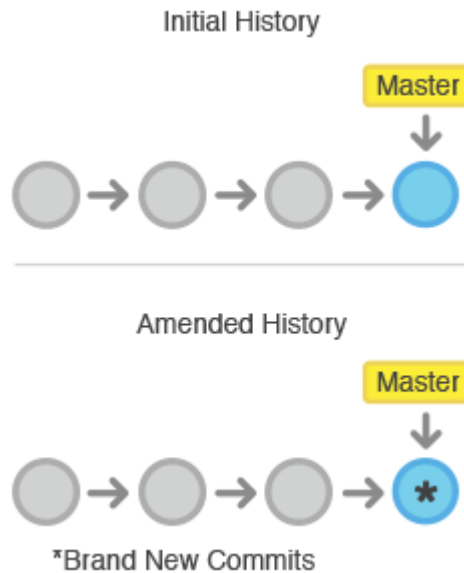
Git check-in/commit

Git commit command is one of the most important git commands. It commits the staged snapshot into the project history. Committed snapshots is something like an original version of the project. This committed snapshot will also act as a backup.

Example: to commit staged snapshot, `git commit` is used. This will launch a text editor prompting user for a commit message. The entered message will be saved and the editor will then create the actual commit: `git commit -m "<message>"`. But to commit a snapshot of all changes of the tracked files in the working directory, use `git commit -a`.

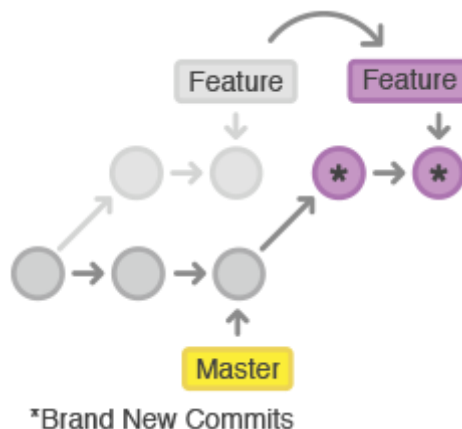
Git Edit

Changing last commit is the most common rewriting history. To change the commit message or the snapshot, use the `git commit --amend` command. This command is a simple way to fix up the most recent commit.



Amending doesn't just alter the recent commit but it also replaces it entirely.

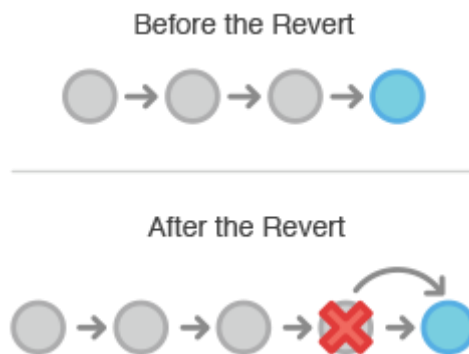
To change multiple message at once, use `git rebase` command. Rebasing is the process of moving a branch to a new base commit.



Running `git rebase -i <base>` begins an interactive rebasing session. Instead of blindly moving all commits to new base, interactive rebasing allows to alter the individual commits during the process. This allow the user to clean up history by removing, splitting and altering the existing series of commit.

Git revert

The `git revert <commit>` command undo the committed snapshot. Instead of removing the commit from the project history, it undo the changes introduced by the commit and appends a new commit. This prevents Git from losing history.



Conflicts

Merge Conflict:

Sometimes when merging or pulling from a branch, user might encounter merge conflicts. Git will then display CONFLICT (content): Merge conflict in Fakefile and request user to fix and commit the result. A merge conflict usually occurs when the current branch and the branch user want to merge into the current branch have diverged. The current branch which are not in other branch is committed, and vice versa. The current branch will be compared with the other branch and if there are disagreeing changes, merge conflict occur. The conflicted file will be called conflict markers.

Resolution for Merge Conflict:

When this sort of conflict occurs, Git writes a special block into the file that contains the contents of both versions where the conflict occurred. To complete this type of merge, use your text editor to resolve the conflict, then add the file and commit it to complete the merge.

Removed file conflict:

Removed file conflicts occur when one person edits a file, and another person deletes that file in their branch. Git does not know if you want to keep the file with the new edits, or delete the file and forget about those edits. This example will show how to resolve this both ways.

Resolution for removed file conflict:

First, to resolve the conflict by keeping the new changes. Suppose that you added a line to your *README.md* in `branch-b`, but someone else has deleted the file entirely in `branch-c`. Conflict then occurred. You can resolve this by adding the file back, and committing it once more.

Git Merge

Git merge is used to join two or more development histories together. This command take independent lines of development created by git branch and integrate them into single branch.

Why merging is necessary.

Most of the time, git merge is necessary for 2 people on 2 different computers modifying the file at the same time. It used to merge the two branches, result in a single collection of files that contains both sets of changes. Git merge is important because it is used to preserve the complete history of the project and prevent the risk of re-writing public commit.

DEMO

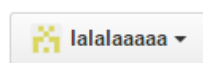
The basic git command will be demonstrated.

To demo the git command, first create a repository.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name



Great repository names are short and memorable. Need inspiration? How about **stunning-sniffle**.

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

After creating the repository, copy the link highlighted in the diagram below to clone in the future.

gtham96 / TestGit

Unwatch ▾ 3 ★ Star 0 🍴 Fork 0

[Code](#) ⓘ Issues 0 🔄 Pull requests 0 📖 Wiki 📡 Pulse 📊 Graphs

No description or website provided.

4 commits 1 branch 0 releases 1 contributor

Branch: master ▾ **New pull request** New file Upload files Find file HTTPS ▾ <https://github.com/gtham96/TestGit> 📄 📄 Download ZIP

lalalaaaaa lalala	Latest commit 1fbc7a3 19 minutes ago	
Test	lol	7 days ago
.gitattributes	add first project	7 days ago
.gitignore	add first project	7 days ago

To clone the repository, first start the git shell, and type in git clone “the link copied”. This command will clone the repository as the local repository. The directory can be found using the ls command and enter the directory using the cd command.

```
C:\Users\presentation\Documents\GitHub> git clone https://github.com/gtham96/TestGit.git
Cloning into 'TestGit'...
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 19 (delta 3), reused 16 (delta 0), pack-reused 0
Unpacking objects: 100% (19/19), done.
Checking connectivity... done.
C:\Users\presentation\Documents\GitHub> ls

Directory: C:\Users\presentation\Documents\GitHub

Mode                LastWriteTime         Length Name
----                -
d-----          5/10/2016   9:36 AM             TestGit

C:\Users\presentation\Documents\GitHub> cd TestGit
C:\Users\presentation\Documents\GitHub\TestGit [master ≡]> ls

Directory: C:\Users\presentation\Documents\GitHub\TestGit

Mode                LastWriteTime         Length Name
----                -
d-----          5/10/2016   9:36 AM             Test
-a---          5/10/2016   9:36 AM           2581 .gitattributes
-a---          5/10/2016   9:36 AM           2389 .gitignore
```

To add a new file, use git add “filename”. This will create a file in the local repository but not the online repository. To save the changes made, use the git commit –m “commit message”. Before committing, user need to git config their global email and name. After that, the changes can be recorded. User can check how many changes or adding made by looking at the [master=+1 ~0 ~0~]. +1 means there is 1 file added.

```
C:\Users\presentation\Documents\GitHub\TestGit [master ≡ +1 ~0 ~0 ~]> git add "laa.txt"
C:\Users\presentation\Documents\GitHub\TestGit [master ≡ +1 ~0 ~0 ~]> git commit -m "lalala"

*** Please tell me who you are.

Run

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

to set your account's default identity.
Omit --global to set the identity only in this repository.

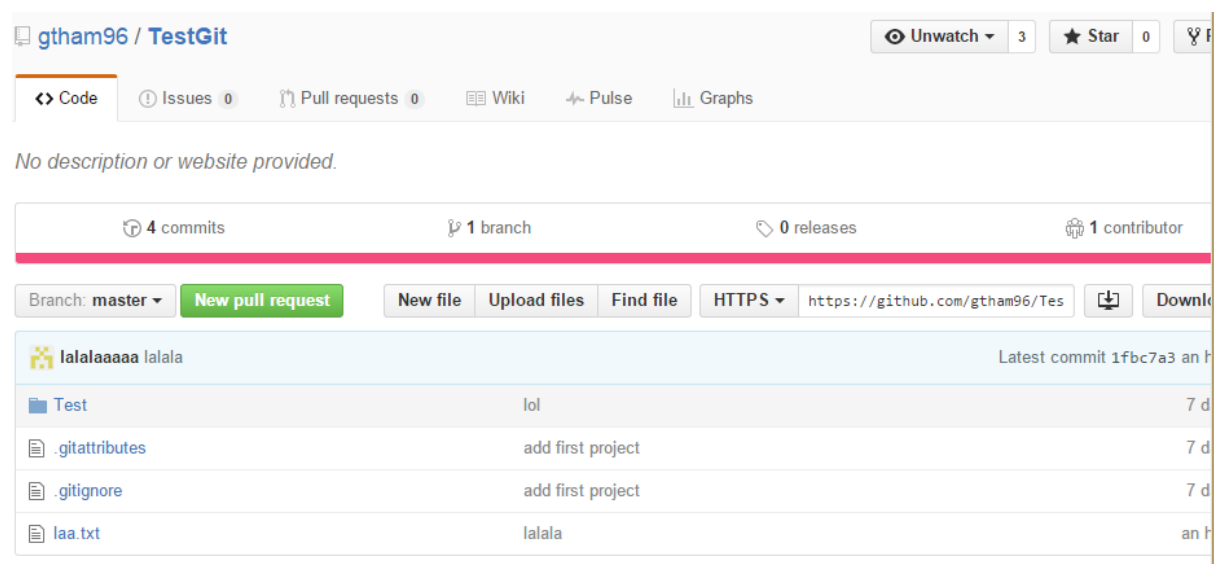
fatal: empty ident name (for <(null)>) not allowed
C:\Users\presentation\Documents\GitHub\TestGit [master ≡ +1 ~0 ~0 ~]> git config --global user.email "nubkia96@gmail.com"
C:\Users\presentation\Documents\GitHub\TestGit [master ≡ +1 ~0 ~0 ~]> git config --global user.name "cs"
C:\Users\presentation\Documents\GitHub\TestGit [master ≡ +1 ~0 ~0 ~]> git commit -m "lalala"
[master 1fbc7a3] lalala
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 laa.txt
```

To apply the changes or add the file use the command git push. This will prompt user to enter their github username and password. After inputting user details, the online repository will apply the changes.

```
C:\Users\presentation\Documents\GitHub\TestGit [master ↑]> git push
Username for 'https://github.com': lalalaaaaa
Password for 'https://lalalaaaaa@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 254 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://github.com/gtham96/TestGit.git
 9885ce9..1fbc7a3 master -> master
C:\Users\presentation\Documents\GitHub\TestGit [master ≡]> ls

Directory: C:\Users\presentation\Documents\GitHub\TestGit

Mode                LastWriteTime         Length Name
----                -
d----          5/10/2016   9:36 AM             Test
-a---          5/10/2016   9:36 AM          2581 .gitattributes
-a---          5/10/2016   9:36 AM          2389 .gitignore
-a---          5/10/2016   9:37 AM             0 laa.txt
```



To retrieve the changes made by other collaborators, type in git pull. If there is any changes made by others, the changes will be applied to the local repository.

```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git pull
Already up-to-date.
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]>
```


To make changes without affecting the master repository, user can create a new branch by using git checkout -b "name". To keep track of the changes of the new branch use git push --set-upstream origin "branch name".

```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git branch
* master
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git checkout -b lalala
Switched to a new branch 'lalala'
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala]> git branch
* lalala
  master
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala]> git push --set-upstream origin lalala
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
 * [new branch]      lalala -> lalala
Branch lalala set up to track remote branch lalala from origin.
```

User can now make changes or add file in the new branch.

```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala ≡]> ls

Directory: C:\Users\presentation\Documents\GitHub\SE-Portfolio-5

Mode                LastWriteTime         Length Name
----                -
-a---             5/10/2016   9:37 AM              0 lalala.txt
-a---             5/10/2016  10:26 AM             16 README.md
-a---             5/10/2016  10:26 AM              0 test.txt


C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala ≡]> git add "branch.txt"
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala ≡ +1 ~0 -0 ~]> git commit -m "add new branch.txt"
[lalala bac8a15] add new branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 branch.txt
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala ≡]> git push
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 230 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
 411b71f..bac8a15  lalala -> lalala
```

To apply the changes from the new branch to the master branch, first git checkout back to the master branch. Then git merge the new branch.

```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [lalala ≡]> git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]> git merge lalala
Updating 411b71f..bac8a15
Fast-forward
 branch.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 branch.txt
```

To apply changes to the online repository simply use git push.

```
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ↑]> git push
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/gtham96/SE-Portfolio-5.git
 411b71f..bac8a15  master -> master
C:\Users\presentation\Documents\GitHub\SE-Portfolio-5 [master ≡]>
```

 **gtham96 / SE-Portfolio-5**

Unwatch 3

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Wiki

Pulse

Graphs

No description or website provided.

4 commits

2 branches

0 releases

1 contributor

Branch: master

New pull request

New file


Upload files





Find file


HTTPS

https://github.com/gtham96/SE-

Download ZIP

 **lalalaaaa** add new branch.txt Latest commit bac8a15 4 minutes ago

 README.md	Initial commit	an hour ago
 branch.txt	add new branch.txt	4 minutes ago
 laa.txt	damn	25 minutes ago
 test.txt	New text file	an hour ago

 README.md