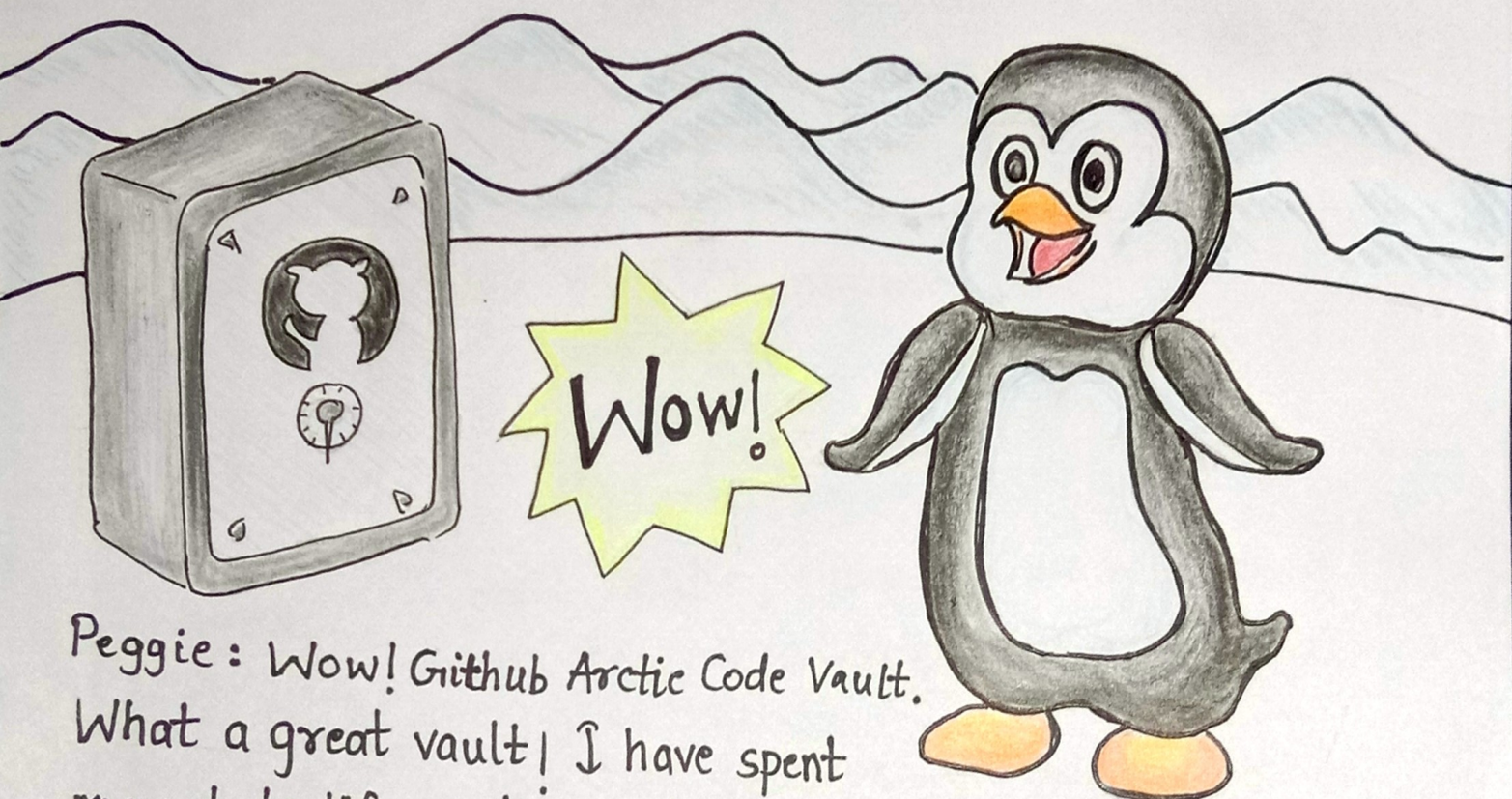
Peggit



Peggie: Wow! Github Arctic Code Vault. What a great vault! I have spent my whole life writing codes and I don't want my work to get wasted. I wish my coming generation could use it. But I know nothing about git ☹.

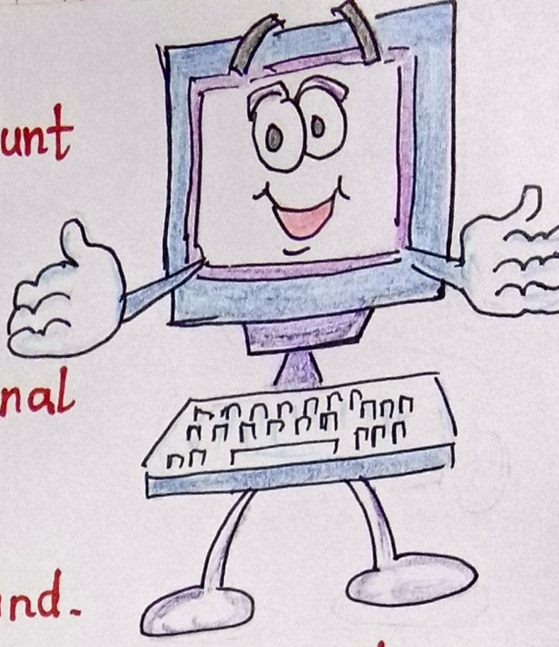
Submitted by:

Aishna Agrawal (18110011)

Janvi Thakkar (18110069)

Table of Contents:

Don't worry Peggie, create an account and hop on!



Let's start with the fundamentals!
Download git bash and open the terminal in the working directory.

We will start by using 'git init' command.
It helps us to create .gitignore files for a new repository.
After this we will add the file to the stagging area by using 'git add [file]' command.

```
$ git init
$ git add [file name]
```

Wait Toddie, I have so many files, I can not keep on adding all the files individually.

For that we have 'git add * ' command. ☺
It helps us to add more than one files together.

Now we will commit our files to record them permanently in the version history.
And so, we will use 'git commit -m "msg" '.

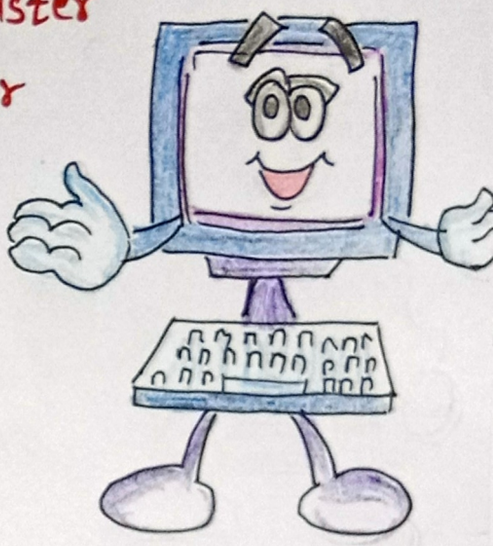
```
$ git add *
$ git commit -m "my children will inherit "
```

Yeahh

This is so easy. What to do next, Toddie?

Moving forward we will use 'git remote add [variable name] [remote server link]' to connect our local repository to the remote server.

The command 'git push [variable name] master' will help you commit the changes to our remote repository.



```
$ git remote add origin "url"
$ git push origin master
```

Oops! Toddie, it is asking for my password. Which password?

The password is same as the password for your git account.

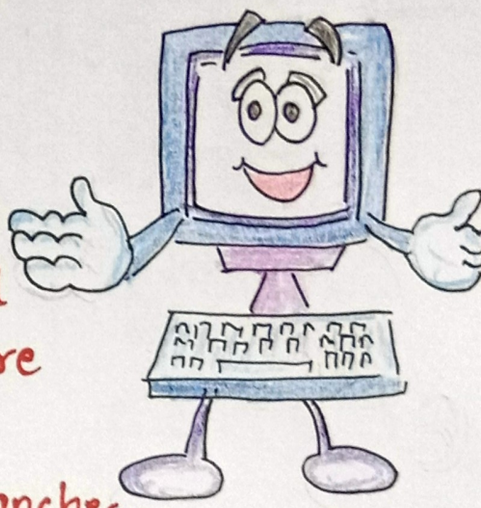
```
$ git config -global user.name "[name]"
$ git config -global user.email "[email address]"
```

- Through these commands you will be able to set the author name and email address respectively to be used with your commits.

Oohh!! nice, this is working.

Toddie, I have done several changes in my code, and I am not sure about it ☹
But I want to save this piece of code as well. Should I create another repository.

No, no, Peggie. Git allows you to create new branch and once you are sure that your code is good to go, git also provide you with the option to merge two branches.



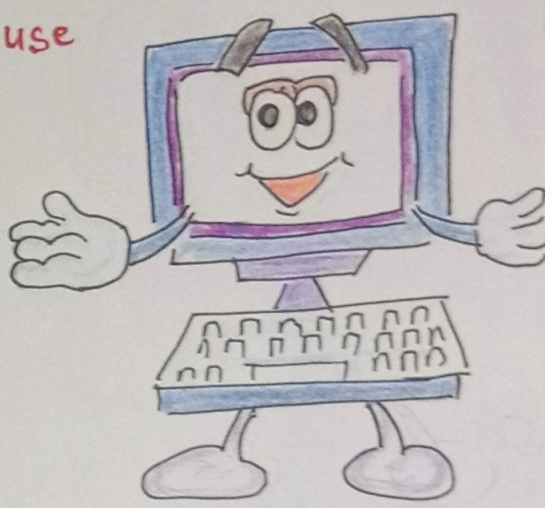
Ohh! That's such a great thing.
Let me explain you,
'git branch [branch name]'- this creates a new branch and now you can add all your files and commit in it, and you are done.
If you want to know about all your branches use the command 'git branch'. This will list out all your branches.

Another important command that you should remember is, 'git checkout [branch name]'. It is used to switch from one branch to another.

'git merge [branch name]' command merges the specified branch history into the current branch.

Toddie, I did something terribly wrong, please tell me git has the ability to time travel. ☹

Yeah! of course Peggie. Just use 'git reflog' It lists everything you have done in git, across all branches! Each one has an index Head@{index}. So now find the one before you broke everything. Now use 'git reset HEAD@{index}. And you are back in time!!



Git is so amazing!!

Toddie, I just want to make a small change, can I do it?

Yes, Peggie. To make your changes use 'git add.', it adds individual file.

'git commit --amend --no-edit'. Through this command your last commit contains the changes.

Toddie, I accidentally committed to the master that should have been on brand new branch! How can I do such a blunder!

Don't worry Peggie! Just create a new branch from the current state of master by using the command;

```
$ git branch [branch name]
$ git reset HEAD~ --hard
$ git checkout [some new branch]
```

And now your commit will live in this branch. :)

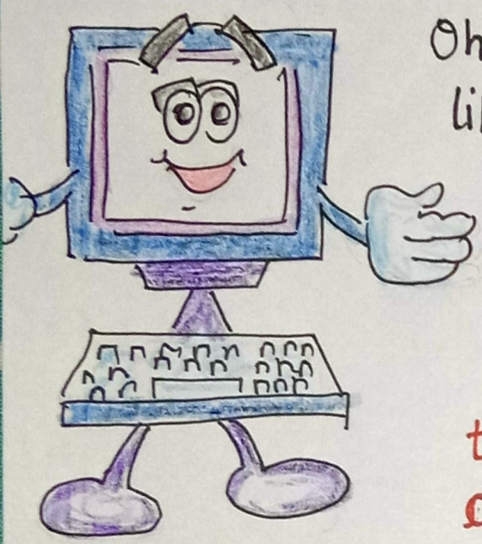
Toddie! I created so many branches and now I don't Know what are the difference between them.

'git diff' shows the file differences which are not yet stage.

Moreover, 'git diff [branch name1] [branch name 2]' shows the difference between two branches mentioned.

Wow Toddie, this is so nice.

Oh no! I want to undo a commit done from like 5 commits ago! What to do now?



Find the commit you need to undo, then,

$ git log

Use the arrow Key to scroll up and down in the history. And once you have found your commit, save the hash by using,

$ git revert [saved hash]

Now git will create a new commit that undoes that commit. Follow the prompt to edit the commit message or just save and commit.

Cool! Finally it is done....

Reference used:
https://www.edureka.co/blog/git-commands-with-example/#git.