Git Introduction

CS 400

February 11, 2018

1 Introduction

Git is one of the most popular version control system. It is a mature, actively maintained open source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel.

2 Installation

Please follow following link for installing git on your system.
https://www.atlassian.com/git/tutorials/install-git

3 Getting Started

3.1 Configuration

You can run below commands with your name and email id to associate all your git repositories with this name and email id.

```
git config --global user.name "your name"
git config --global user.email "your email id"
```

3.2 Initialization

To initialize a Git repository, go to your directory where you want to edit/create your files. Then use the following command:

```
git init
```

Now all the files that you create within this directory can be version controlled using some simple commands. You will learn these commmands below one by one.

3.3 Checking the status

To see what the current state of our project is, use:

```
git status
```

Current status means which files you have modified since your last commit and which files are in the staging areas, ready to be committed. (You will learn about staging are and commit in further sections.)

3.4 Adding changes

To tell Git to start tracking changes made to your files (by committing them), we first need to add it to the staging area by using the following command:

```
git add filename.txt
```

You can think of staged files as files that are ready to be committed and git has been made aware of this.

The staging area is essentially a "holding area" for changes that will be committed when you next do *git commit*.

Staging helps you split up one large change (that you want to make to your source codes,especially in multiple files) into multiple independent commits.

3.5 Committing

To store our staged changes we run the following command with a message describing what we've changed.

```
git commit -m "Add comment here"
```

You should add meaningful comments. When you go back to some earlier version of your code, comments will help you in tracking the changes efficiently.

Example:

```
git commit -m "Implemented insertion and deletion"
```

Committed means that the data is safely stored in your local database. You should ideally commit any important conceptual changes in your files/source code.

Following figure indicates three areas (conceptual) where your files are stored.

3.6 Adding all changes

You can add all the new files or multiple files using following commands.

```
git add *.txt
```

Above command will add all the files with "txt" extension.

```
git add .
```

Last command will add all the files in your working directory to the staging area.

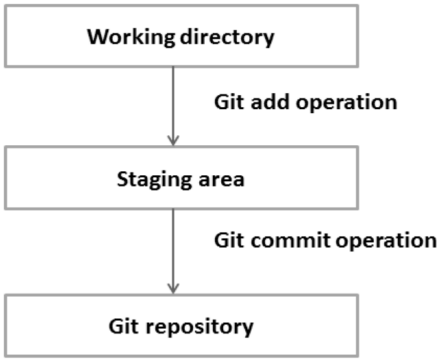

Figure 1: Git terminologies

3.7 History

To track all the changes you have committed so far, you can use the following command:

```
git log
```

```
Sonus-MacBook-Pro:GitDemo1 sonuagarwal$ git log
commit d28839bf281bb392921c2d42a31b4c74c7e89e56 (HEAD -> master)
Author: Sonu Agarwal <sonuagrwal18@gmail.com>
Date:   Sun Feb 11 21:28:48 2018 -0600

    added user input functionlity in hello world program

commit 95f2f69d276fef6b680fa3bbcc411963606f9172
Author: Sonu Agarwal <sonuagrwal18@gmail.com>
Date:   Sun Feb 11 21:26:27 2018 -0600

    added hello world program
Sonus-MacBook-Pro:GitDemo1 sonuagarwal$
```

Figure 2: Example output of "git log" command

3.8 Remote Repositories

Till now all the files your were saved on the local machine. We call it local repository. You can also save the exact copy of files (with all commit history) at any remote location. We call that remote repository. Github is a free platform where you can save your repository.

To push our local repo to the GitHub server we'll need to add a remote repository by using following command.

```
git remote add origin repositoryLink
```

Example:

```
git remote add origin git@github.com:sonuagr/GitDemo1.git
```

'origin' is the name of the remote. You can use any name, but origin is the conventionally used name. You will get the repository link once you create a new repository on Github.

3.9 Pushing Remotely

Below command tells Git where to push our commits.

```
git push -u origin master
```

The -u tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do. 'origin' is the name of the remote, you want to push your repository to. 'master' is the default name of the local branch.

3.10 Pulling Remotely

After you have pushed your files to the remote repository, your collaborators can make changes to those files and commit/push those changes. To get the latest copies of the files to your local repositories, you can use following command:

```
git pull origin master
```

3.11 Differences

After you have made some changes in your working area, you might want to know what are the changes you did after your last commit. You can do so using below command:

```
git diff HEAD
```

Note: The HEAD is a pointer that holds your position within all your different commits. By default HEAD points to your most recent commit, so it can be used as a quick way to reference that commit.

To view difference between HEAD and staging area, use:

```
git diff --staged
```

You can also view the differences between Stage and working directory using:

```
git diff
```

3.12 Resetting the stage

If you want to remove any file from the staging area i.e., unstage the file, you can use:

```
git reset filename.txt
```

3.13 Undo

Files can be changed back to how they were at the last commit by using the command:

```
git checkout -- filename.txt
```

3.14 Branching out

You can create a new branch anytime to experiment with your files. To create a new branch, use the following command.

```
git branch expt1branch
```

3.15 Switching branches

When you create a new branch, you do not automatically switch to new branch. You are still at the previously checked-out branch.

You need to use following command to switch to a new branch.

```
git checkout expt1branch
```

3.16 Switching back to master

You can switch back to the master branch (default branch when you initialize a git directory) using below command. This is similar to checking out any branch.

```
git checkout master
```

3.17 Preparing to merge

You can merge any two branches. To merge your master branch with the new branch, use following command.

```
git merge checkout expt1branch
```

Note that master is the checked out branch at this moment. Basically, above command will merge the 'new_branch_name' branch with currently checked out branch. Also, 'new_branch_name' will be merged into the 'master' branch.

Automatic merging will occur only if there are no conflicts between the two branched. Otherwise you will get warning, and you will have to resolve the conflicts manually.

3.18 Keep Things Clean

To delete any branch, use below command.

```
git branch -d expt1branch
```

4 References

You can also use following resources to learn basics of git and github.

1. https://try.github.io/levels/1/challenges/1
2. https://www.youtube.com/watch?v=SWYqp7iY_Tc
3. udacity.com (Course Name: How to Use Git and GitHub?)

This tutorial has been created by Sonu Agarwal for CS 400 course. You can send email at sonu@cs.wisc.edu for any clarification or reporting any error.