01-Git | Introductions

VCS, Git, Git hosting, Installation

Posted on February 21, 2021
Last updated on February 20, 2023

ALL Pdf Notes (https://github.com/amrit94/DataScience/tree/main/notes)

1.1 What is version control?

- Version control is a management system that allows you to `record and track changes` in your source code and files so that you can recall certain versions later.
- It's like a Google Doc for programming, where you can collaborate with multiple people working on the same code and `see the source code's history`

Ultimately, using a version control system allows teams to streamline their development process, which `improves their overall efficiency`

1.2 Why is it useful?

Tracking: Say you are working on a web application, and one day, you find that your code changes have broken parts of the website. Instead of going through the trouble of finding the bug, you can `revert your changes` and see what lines of code are causing the problem.

Teamwork: Without a version control system, it's `challenging to work on the same source code at once`. By using something like Git, you can more easily merge changes, which makes it significantly `easier to collaborate on projects`.

Branches: Say you are working on the footer and header of your website without using a version control system. You've finished the header, but not the footer, which means that the project isn't ready to be public. With a version control system, you can `create branches for different aspects of the project` you are working on and `merge them into the main source code when you are done`.

1.3 Why should I learn version control?

Due to how useful version control is, it's almost always a `requirement for any developer` or engineering job. As you become more accustomed to using a version control system, you will realize how powerful and easy it is to use.

Popular `version control systems`:

- Github
- GitLab
- Bitbucket
- Beanstalk
- PerForce

1.4 What is git?

Git is a `free`, `open-source` distributed version control system. It keeps track of projects and files as they change over time with the help of different contributors.

Version control is a system that records changes to a file, or set of files, over time so that you can recall specific versions later.

Git helps keep track of changes made to a code. If at any point during coding you `hit a fatal error` and don't know what's causing it, Git allows you to `revert back` to a stable state. It also helps you `see what changes have been made to` the code over time.

Repository

A repository (commonly referred to as repo) is a collection of source code. A repository has commits to the project or a set of references to the commits (i.e., heads).

Commits

A commit `logs a change` or series of changes that you have made to a file in the repository. A commit has a unique SHA1 hash which is used to keep track of files changed in the past. A series of commits comprises the `Git history`.

Branches

A branch is essentially a unique set of code changes with a unique name. Each repository can have `one or more branches`. The main branch — the branch where all the changes `eventually get merged into - is called the master`. This is the official working version of your project and the one that you will see when you visit the project repository at github.com/yourname/projectname.

Working directory, staging area, and local repo

Every local repository has three different virtual zones. These are:

- Working directory
- Staging area
- Commit area

The **working directory** is where new `files are created`, old `files are deleted`, or where `changes are made` to already existing files.

Once changes are made, they are added to the **staging area**. The staging area is also `sometimes called the index`.

Once the changes are complete, the staging area will contain one or more files that need to be committed. Creating a commit will cause Git to take the new code from the staging area and make the commit to the main repository. This commit is then moved to the **commit area**.

1.5 Create a Gitlab and Github Account

Gitlab

GitLab is a popular, open source Git hosting solution implemented by more than 50,000 organizations. Over the last few years, GitLab has evolved with strong community support and growth, handling thousands of users on a single server and several such servers on an active cluster.

Github

GitHub is a website where millions of developers gather every day to collaborate on open-source software. It's also the place that hosts billions of lines of code, and also a place where users of software go to report issues they might have.

In short, it's a platform for software developers, and it's built around Git.

As a developer, you can't avoid using GitHub daily, either to host your code or to make use of other people's code. This post explains you some key concepts of GitHub, and how to use some of its features that improve your workflow, and how to integrate other applications into your process. `

1.6 How to install git?

Check If Git is Installed

Some operating systems also come with Git installed, so it is worth checking before you install. You can check whether Git is installed and what version you are using by `opening up a terminal window in Linux or Mac, or a command prompt window in Windows`, and typing the following command:

```
git --version
```

However, if Git is not installed, you will receive an error similar to the following:

```
-bash: git: command not found

'git' is not recognized as an internal or external command,
operable program or batch file.
```

In this case, you should install Git into your machine. Let's go through installation for several of the major operating systems.

Installing Git on Linux

By far the easiest way of getting Git installed and ready to use is by using your version of Linux's default repositories. Let's go through how to install Git on your local Linux machine using this method.

Installing Git on Ubuntu 18.04 or Debian 10 You can use the apt package management tools to update your local package index. Afterwards, you can download and install the program:

```
sudo apt update
sudo apt install git
```

For more information, read this

https://www.digitalocean.com/community/tutorials/how-to-contribute-to-open-source-getting-started-with-git (https://www.digitalocean.com/community/tutorials/how-to-contribute-to-open-source-getting-started-with-git)

1.7 Setting up Git

Now that you have Git installed, you need to do a few things so that the commit messages that will be generated for you will contain your `correct information`.

The easiest way of doing this is through the git config command. Specifically, we need to provide our name and email address because Git embeds this information into each commit we do. We can go ahead and add this information by typing:

```
git config --global user.name "Your Name"
git config --global user.email "youremail@domain.com"
```

git configuration file

We can see all of the configuration items that have been set by typing: `git config --list`

```
# git config --list

user.name=Your Name
user.email=youremail@domain.com
```

~/.gitconfig contents

As you can see, this has a slightly different format. The information is stored in your Git configuration file, which you can optionally edit by hand with a text editor, like nano for example: `vim ~/.gitconfig`

```
# vim ~/.gitconfig

[user]
    name = Your Name
    email = youremail@domain.com
```

Once you're done editing your file, you can exit nano by typing the `Ctrl + x` keys, and when prompted to save the file press `y`.

There are many other options that you can set, but these are the two essential ones needed to prevent warnings in the future.