**Version control system**

System that is use to manages files , directories and the changes made to them, over time. This allows you to recover older versions of your data or examine the history of how your data changed (Version Control with Subversion , Ben Collins-Sussman etl ,2011) .Such system is useful in development process by make it easy to track change in the code and quickly rollback to early releases in case of bad release found .here is some VCS feature :-

1. Provide share repository that developer or group of developers can use it to share their codes.
2. Each developer can make his own change and easy uploaded to the repository.
3. Also track the change and provide history that shown change over time.
4. Roll backing from bad releases are also possible.
5. Branches concept to breakdown the code and store related parts of code together in one branch.

**Common Version control system**

There are many VC software which provide all this feature with some different in implementation such as :

* **Git version control** :

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency(https://git-scm.com).

**1.Branching and Merging:-**

The Git feature that really makes it stand apart from nearly every other SCM out there is its branching model(https://git-scm.com/about).

Git allows and encourages you to have multiple local branches that can be entirely independent of each other. The creation, merging, and deletion of those lines of development takes seconds(https://git-scm.com/about).

This means that you can do things like:

* 1. **Frictionless Context Switching**. Create a branch to try out an idea, commit a few times, switch back to where you branched from, apply a patch, switch back to where you are experimenting, and merge it in(https://git-scm.com/about).
  2. **Role-Based Codelines**. Have a branch that always contains only what goes to production, another that you merge work into for testing, and several smaller ones for day to day work(https://git-scm.com/about).
  3. **Feature Based Workflow**. Create new branches for each new feature you're working on so you can seamlessly switch back and forth between them, then delete each branch when that feature gets merged into your main line(<https://git-scm.com/about>).
  4. **Disposable Experimentation**. Create a branch to experiment in, realize it's not going to work, and just delete it - abandoning the work—with nobody else ever seeing it (even if you've pushed other branches in the meantime(<https://git-scm.com/about>). Notably, when you push to a remote repository, you do not have to push all of your branches. You can choose to share just one of your branches, a few of them, or all of them. This tends to free people to try new ideas without worrying about having to plan how and when they are going to merge it in or share it with others(<https://git-scm.com/about>).

**2.Small and fast :**

**Git is fast**. With Git, nearly all operations are performed locally, giving it a huge speed advantage on centralized systems that constantly have to communicate with a server somewhere(<https://git-scm.com/about>).

Git was built to work on the Linux kernel, meaning that it has had to effectively handle large repositories from day one. Git is written in C, reducing the overhead of runtimes associated with higher-level languages. Speed and performance has been a primary design goal of the Git from the start(<https://git-scm.com/about>).

**3.Distributed:-**

One of the nicest features of any Distributed SCM, Git included, is that it's distributed. This means that instead of doing a "checkout" of the current tip of the source code, you do a "clone" of the entire repository((<https://git-scm.com/about>).

**Multiple Backups**

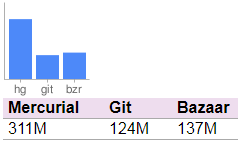
This means that even if you're using a centralized workflow, every user essentially has a full backup of the main server. Each of these copies could be pushed up to replace the main server in the event of a crash or corruption. In effect, there is no single point of failure with Git unless there is only a single copy of the repository((<https://git-scm.com/about>).

**4.Data Assurance** :-

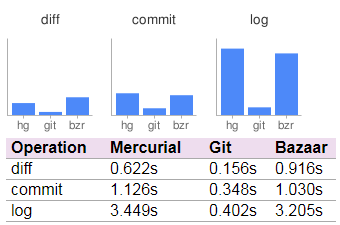
The data model that Git uses ensures the cryptographic integrity of every bit of your project. Every file and commit is checksummed and retrieved by its checksum when checked back out. It's impossible to get anything out of Git other than the **exact bits you put in**.

* **Bazaar version control :**

Bazaar is a version control system that helps you track project history over time and to collaborate easily with others. Whether you're a single developer, a co-located team or a community of developers scattered across the world, Bazaar scales and adapts to meet your needs(<http://bazaar.canonical.com/en/>).We take great pride in making Bazaar easy to learn, easy to use and suitable for everyone (<http://bazaar.canonical.com/en/>).Like any of the distributed VCS tools, one of the coolest features of Bazaar is that it’s distributed. This means that instead of having just the latest revision of the source code locally, you get a complete copy of the history for the branch .This means that you can work while disconnected from a remote server (<http://bazaar.canonical.com/en/>) .Also Bazaar is cross platform While Linux and Mac OS X are hugely important platforms for software developers, the majority of people - developers included - still run on Windows.



Comparing between mercurial , Git , Bazaar in repository size(figure 1)



Comparing between mercurial , Git , Bazaar in common command(diff , commit , log ) speed (figure 2)

**Referances** :-

* <https://bazaar.canonical.com> , bazaar website ,seen in 27/3
* <https://git-scm.com/about> , Git website , seen in 27/3
* Version Control with Subversion , by Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato,2011