



## **Week4-5**

# **Using Project Tools**



# Using Project Tools

- ▣ Collaboration Tools
  - Asynchronous Communication
  - Synchronous Communication
  - Shared Documents





# Collaboration Tools

Working with the *code base* as a member of a development team requires *special tools* and *skills*, many of which are not needed when working on a single programming task in isolation.

Any collaborative process requires an effective way to communicate among the members about their individual activities. They need *shared access* to the code base they are constructing.

The developer tools can be divided into two broad categories:

- ☐ collaboration tools: This is about strategies for working as a team
- ☐ code management tools: This is about strategies for organizing and working with the code base.



Basic and regular collaborative tools used by developers of a typical FOSS project include the followings

- A Web site for public outreach
  - Mailing lists for team communication
  - Real-time chat (IRC) channels
  - Version control to manage the code base
  - Bug tracking tools
- } Interpersonal communication
- } Interaction with code

## Asynchronous Communication

The most versatile collaborative tool, one seen in almost every FOSS project, is the mailing list. A mailing list is a distribution list in which a mail from any member is distributed to all other list members there by supporting *asynchronous communication* within the project's development community.

For many projects different *lists and sub-lists* can be set up to support different aspects of the project e.g., coding, documentation, and management.



# Example:

The **Sahana project** maintains the following lists

- The *Sahana-user list* enables users and administrators of Sahana to submit queries about using, deploying, and administering the software.
- The *Humanitarian-ICT list* serves non-technical members of the community who specialize in emergency management.
- The **Sahana-maindev list** is for software developers and focuses on technical design, quality assurance, documentation, deployment, and geeky discussions about the Sahana code base.





Other forms of asynchronous communication tools include:

***Forums*** which can be used to manage group discussions for a project, forum is topic based, it is particularly well with the dynamics of FOSS project development.

Unlike mailing lists, forums require members to visit a certain Web site in order to participate.

Many tools are available that support asynchronous communication in this way. We recommend ones that are either free or inexpensive, multi-platform, and multi-functional, such as Google Groups (<http://groups.google.com>) and Basecamp (<http://www.basecamphq.com>) or AnyDesk (<https://anydesk.com/en>)



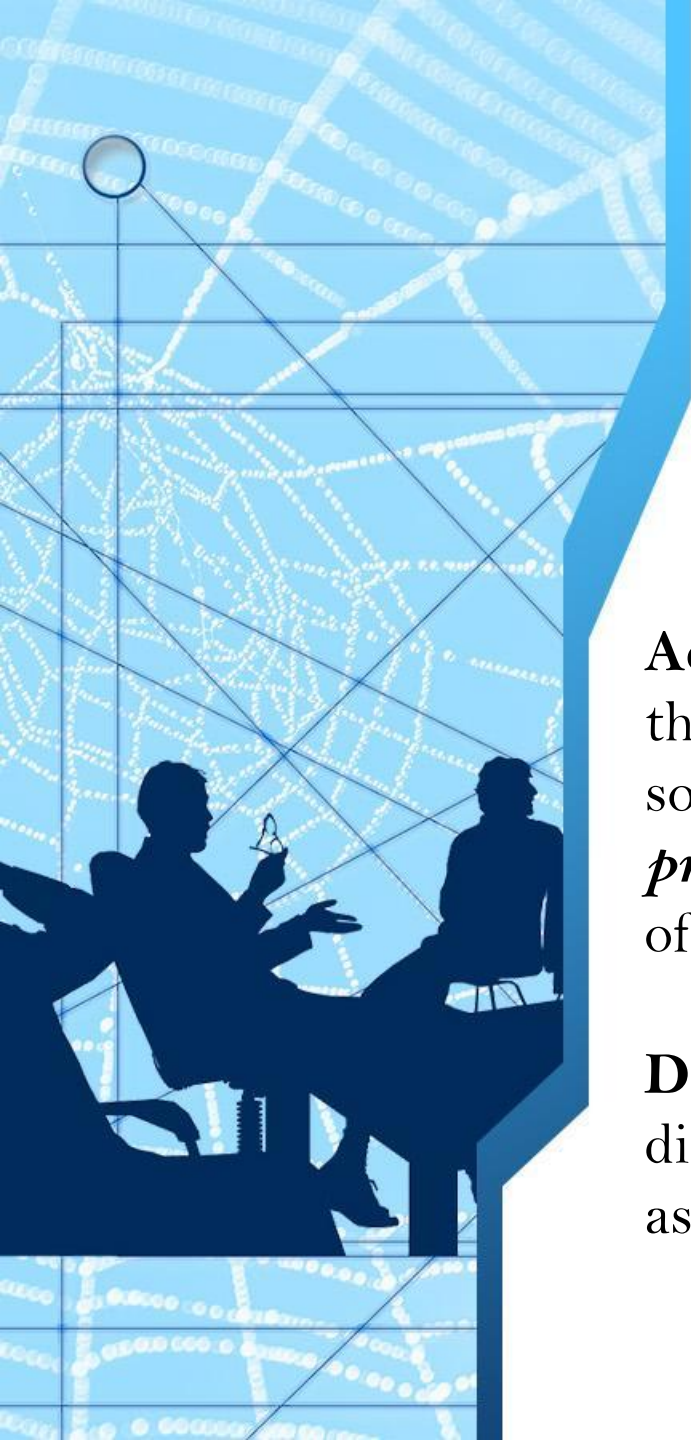
## Synchronous Communication

In today's software development world the members of a software team are geographically dispersed. To accommodate this **constraint** and ensure *continuous communication*, the team must maintain a regular schedule of synchronized communication.

Virtual meetings are sometimes needed to set *goals*, *evaluate progress*, or *discuss some other global issue*

A classical synchronous communication tool is **Internet Relay Chat (IRC)**, which provides real-time text communication over the Internet to support synchronous conferencing.

Many different freely licensed IRC clients are available  
(see <http://ircreviews.org/clients>).



**Videoconferencing** is a more sophisticated tool for managing synchronized communication. It has the advantage of allowing participants to share information visually, as well as through the use of audio and text

**Tools include:** ZOOM, Google meet, Skype, AIM/iChat(<http://aim.com>) and others

**Advantage:** The use of synchronized communication is a critical activity through which software team members can **negotiate** and *set new goals, assess their progress,* and *discuss design and implementation* alternatives at a high level of abstraction.

**Disadvantage:** Synchronized communication is not usually the best place for discussing detailed *technical issues* or *bug fixes*; discussion forums or other asynchronous methods can be used for such





# Shared Documents

In a collaborative software project maintaining shared documents such as to-do lists, milestones, and files is vital

**To-Do Lists:** are lists of tasks to be completed in the near term, along with the names of team members who have committed to complete those tasks. Bug-fixing assignment is a particularly popular use of to-do lists.

**Milestones:** are calendar events that identify deadlines for completing major steps in the project.

**Files:** are artifacts like design documents, client-provided scenarios, community outreach documents, or help pages that support the project's development.


## Code Management Tools

Items to discuss here include the following

- i. The IDE
- ii. The Software Stack
- iii. The Version Control System
- iv. The Bug Tracker

i).**IDE:** An integrated development environment (IDE) is *a software application* that supports the development of software in a particular programming language. This activity includes *coding, compiling, tracing, debugging, and running individual programs* that comprise the code base for a software system.

**Examples of IDEs** include Eclipse, NetBeans, Microsoft Visual Studio, Xcode and many others more.



**Note:** All of these are multi-language IDEs, so that any one can be configured to support development in a particular programming language

Each team member with installed IDE has access to the following kinds of support tools:

- ***Source code editor**, typically a text editor that has such features as syntax highlighting, syntax checking, autocompletion of language-specific terms, and others that simplify the coding process*
- ***Source code translators**, including compilers and interpreters depending on the programming language being used*
- ***Package build tools** that link programmer-developed source code with binaries, libraries, and other resources that make up an application*
- ***Debugging and unit testing tools** that assist the programmer in finding and fixing run-time errors*
- ***Version control tools** that help to synchronize the code base with the repository*
- ***Documentation tools** that support the maintenance of consistent documentation of individual modules and functions in the code base*





## ii).The Software Stack

To contribute effectively to a software project, a developer must set up a **run-time environment** for the code base so that new code can be immediately tested as it is written.

This environment is called a software stack since it is a hierarchy.

An application stack consist of

- A particular operating system,
- A Web Server
- Database Management System
- A programming language that will be used to develop and support the application.

**Examples include:** WAMP, LAMP, WISA,MEAN and others

## iii).The Version Control System

Software systems are made up of a large number of les that change over time as additions, deletions, and modifications are made to the code base. For all but the very smallest projects, it is essential to use a version control system (VCS) to help manage this process.



A VCS (also known as **revision control or source control**) is software that helps manage changes to a collection of source code files.

A VCS keep tracks of all changes to the code base. That means that whenever a programmer adds a new piece of code to the code base, it doesn't simply overwrite the previous version. It also keeps track of who contributed the code, when it was contributed, and so forth.

A VCS supports the following code management functions

- a. Multiple revisions of the same code base
- b. Multiple concurrent developers
- c. Locking, synchronization, and concurrency of the code base
- d. Support for versioning history and project forking, meaning that certain parts of a project or the entire project itself can be split off (forked) into a separate project.

**Example:** Wikipedia(non-software):-It is possible to examine the history of the page, which will show exactly what changes were made, by whom they were made, and when they were made.



## VCS concepts and operations supporting the Management Functions

**Repository:** The repository is the place where the shared code base is stored from where authorized programmer can download and read code.

**Working copy:** This refers to the individual programmer's copy of the code base.

**Check out:** Individual programmers can check out a working copy of the code base, downloading it to his or her computer

**Update:** The programmer can download the most recent version of a code base that was previously checked out. The updated version will include changes made by other programmers.

**Commit:** Programmers who have sufficient write privileges with the repository can contribute new code directly.





**Merge:** Two pieces of code are merged whenever they both apply to the same file or code segment.

**Patch:** A patch is piece of code that is added to the code base. It usually describes what code is to be added or removed from the code base.

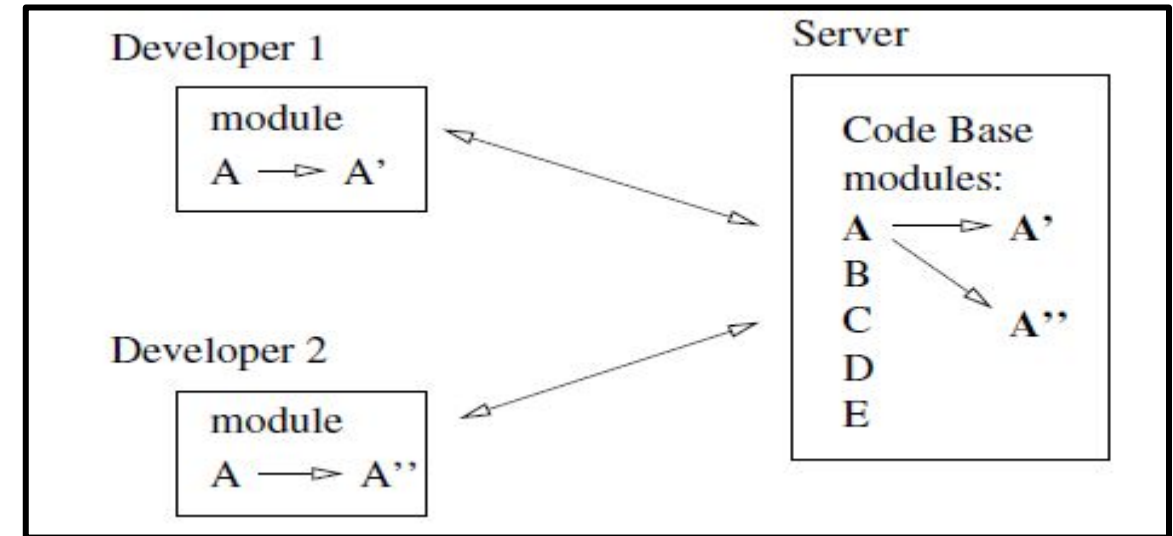
**Trunk:** Repositories are organized into hierarchies, which are tree structures. The trunk is usually the main development line for a project.

**Branch:** A branch is a copy of the software that can undergo separate and independent development. Branches are often used to package releases of the software or to allow experimentation. In some cases a branch can break off (or fork) into a separate or competing project. In many cases, branches are later merged back into the trunk.

## ■ VCS Conflict Management

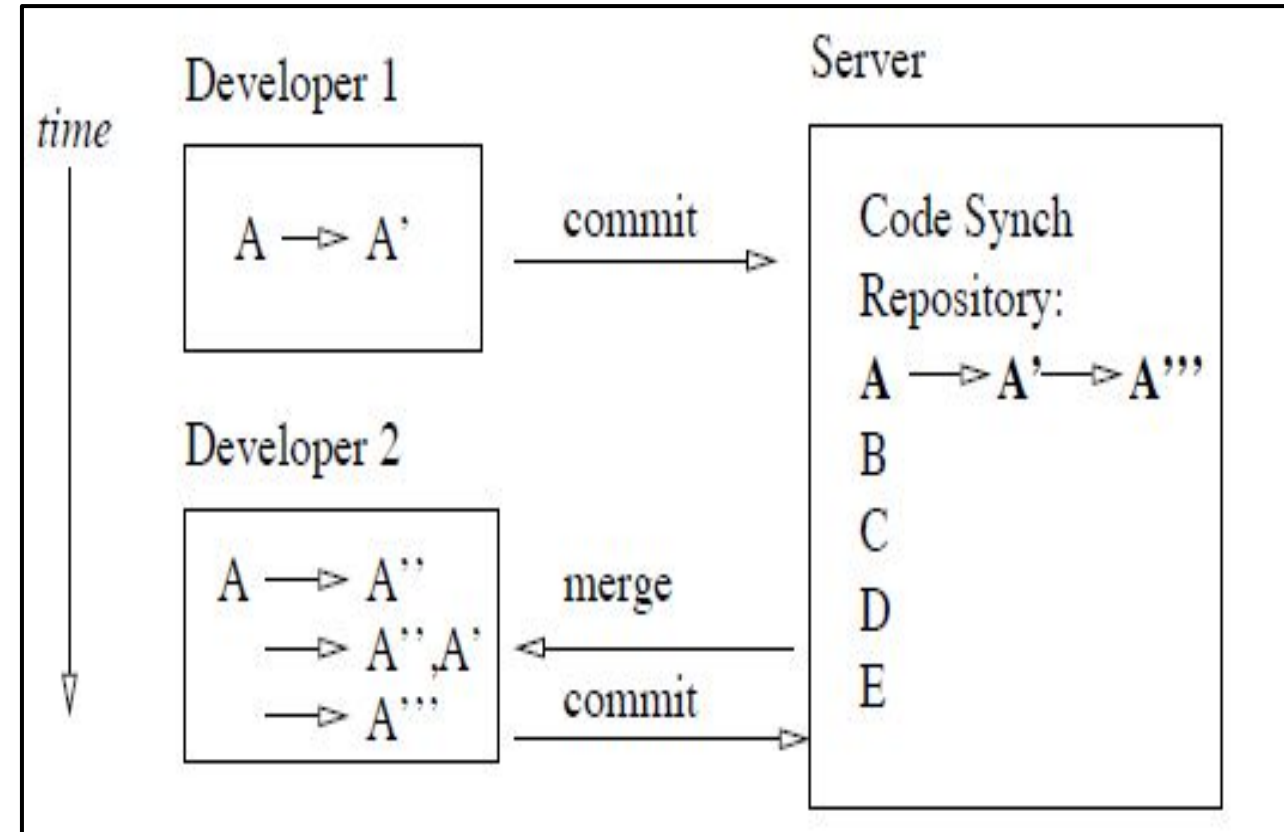
While a VCS helps manage the code base, it does not automatically resolve all possible conflicts that can arise during a collaborative project

Illustration -----□



**Question:** Which variation finally replaces A in the code repository?

The use of a VCS prevents this problem



Conflict Resolution: *Copy-modify-merge*.

Sometimes a conversation between the two developers is needed to resolve this kind of conflict.

In some cases, it may be strategic for one developer to obtain *a temporary and exclusive lock on a module*. Until the lock is released, no other developer can make changes to that module.





## ■ Managing a Code Repository

Collaborative software development is a little like the collaborative encyclopedia building in the Wikipedia project.

The Wikipedia code base i.e., a Wikipedia page is completely open for anyone to read or write. This is not the case for most FOSS development projects.

Not everyone in a community-oriented FOSS project has privileges to write new code directly to the code base

In a *community-oriented FOSS project* everybody has *read access* but only a select few have *write access* and even fewer still have *admin access*



# The Bug Tracker

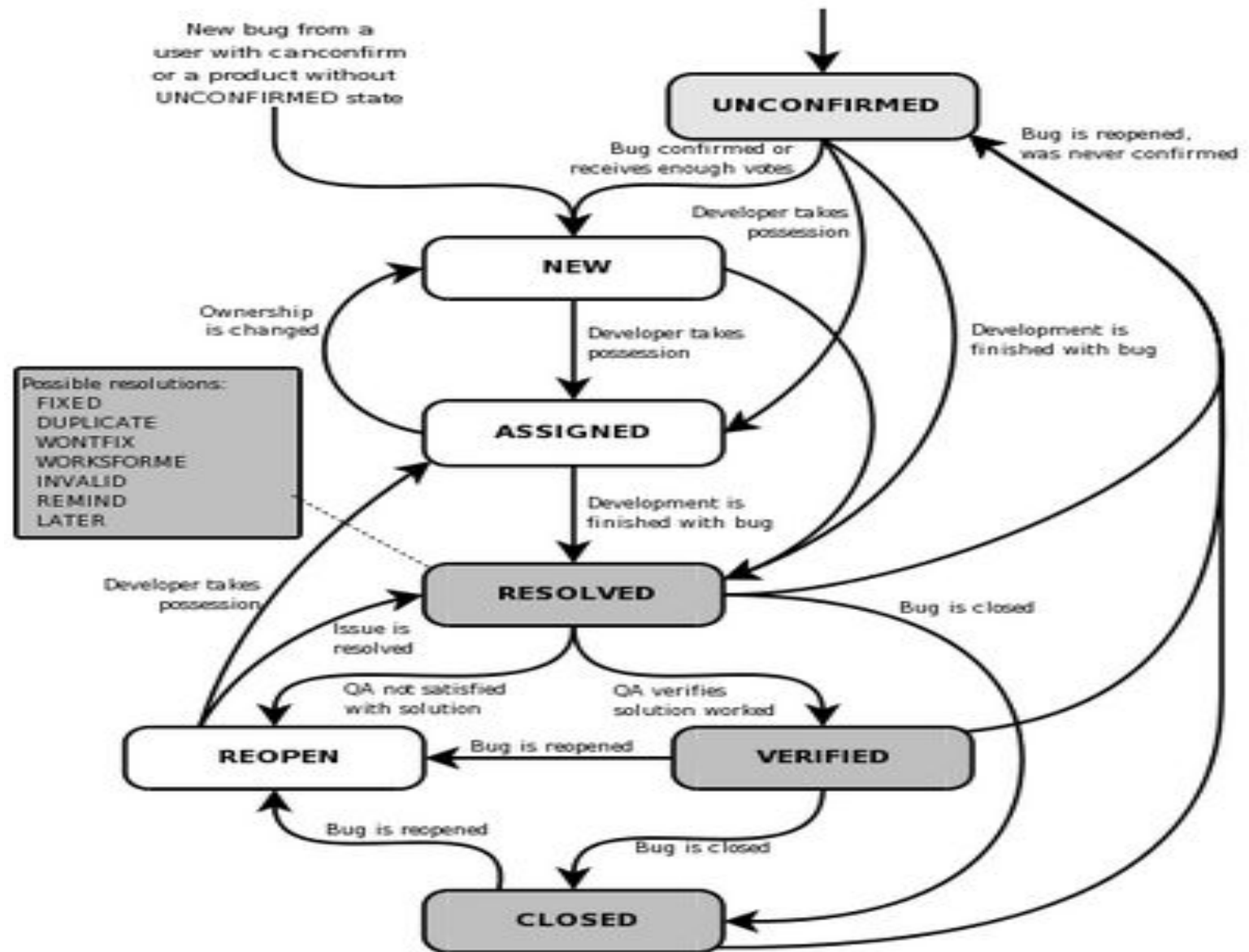
## What is bug tracking?

It is a formalized process established by a software project that governs how *bugs are identified* and *how their resolution is managed* by the developers who work with the code base.

A particularly interesting bug tracking process is the one established by the Mozilla Foundation, based on its open source bug-management tool called Bugzilla (*see <http://www.bugzilla.org/about>*).

## What is “life cycle of a bug”?

This a bug's progress through a series of discrete states. This can be described in the form of a state diagram, as shown below for Bugzilla  
[www.bugzilla.org/docs/2.18/html/](http://www.bugzilla.org/docs/2.18/html/lifecycle.html) lifecycle.html




Lifecycle of a Bugzilla Bug (source: <http://www.bugzilla.org/docs/2.18/html/lifecycle.html>).





# Attributes of a Bug

- **Product and Component** Bugs are divided up by Product and Component, with a Product having one or more Components in it.
  - **Status and Resolution** These define what state the bug is in from unconfirmed to fixed.
  - **Assigned To** The person responsible for fixing the bug.
- \***URL** A URL associated with the bug.
- **Summary** A one-sentence summary of the problem.
- \***Whiteboard** A free-form text area for adding short notes to a bug.
- \***Keywords** The administrator can define keywords used to tag and categorize a bug.
- **Platform and OS** The computing environment where the bug was found.
  - **Version** Version of a product that the particular bug report is about.
  - **Priority** The person assigning the bug sets its priority.



**Severity** Severity of the problem from blocker (application unusable) to trivial (minor cosmetic issue). Also may be used to indicate whether a bug is an enhancement request.

\***Target** (a.k.a. Target Milestone) A future version by which the bug is to be fixed.

**Reporter** The person who led the bug.

**CC list** A list of people who get mail when any of these attributes change.

\***Time Tracking** This form can be used for time tracking.

**Attachments** You can attach files (e.g., test cases or patches) to bugs. If there are any attachments, they are listed in this section.

\***Dependencies** If this bug cannot be fixed unless other bugs are fixed (depends on), or this bug stops other bugs being fixed (blocks), their numbers are recorded here.

\***Votes** Whether this bug has any votes.

**Additional Comments** You can add your two cents to the bug discussion here, if you have something worthwhile to say.



## Run-Time System Constraints

Constraints to development of an effective software application include the limitations of the existing code base, hardware, networks, budgets, and other resources that are available in settings where the software will eventually be used.

### Major constraints by Developers.

#### ☐ Performance-Performance Goals

- **User Interface-should** be consistent and compatible with modern Web-based applications
- On-line help
- Availability and reliability
- Backup and recovery
- Error correction



# Web Hosting

Web hosting is a service that allows organizations and individuals to post a website or web page onto the Internet. A web host, or web hosting service provider, is a business that provides the technologies and services needed for the website or webpage to be viewed in the Internet. Web hosting features are as shown below





# Some Web Hosting Providers

List from CNET 2020 can be provided as follows  
as

- DreamHost
- Bluehost
- A2Hosting
- HostGator
- InMotion Hosting
- Hostinger

## Git and GitHub

**What Is Git?**

**Git** is a version control system. More specifically, Git is a distributed version control system, which means that everyone working with a project in Git has a copy of the full history of the project, not just the current state of the files.



# What Is GitHub?

**GitHub** is a website where you can upload a copy of your Git repository. It allows you to collaborate much more easily with other people on a project. It does that by providing a centralized location to share the repository, a web-based interface to view it, and features like forking, pull requests, issues, and wikis, which allow you to specify, discuss, and review changes with your team more effectively.

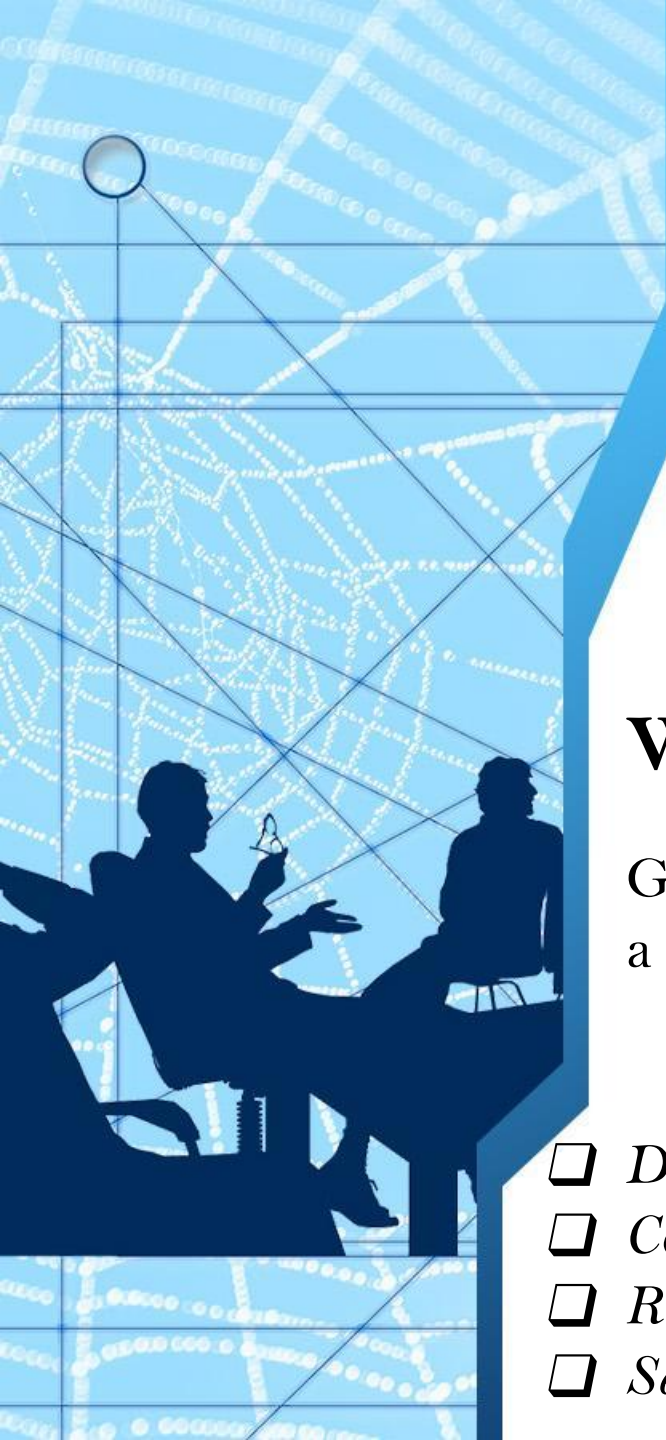
## Why Use Git?

Even if you're working on your own, if you are editing text files, there are a number of benefits to using Git.

### Benefits of Using Gits

- ☐ The ability to undo changes
- ☐ **A complete history of all the changes**
- ☐ **Documentation of why changes were made**



- 
- ☐ *The confidence to change anything*
  - ☐ *Multiple streams of history*

### Benefits when working with a Team

- ☐ *The ability to resolve conflicts*
- ☐ *Independent streams of history*

## Why Use GitHub?

GitHub is much more than just a place to store your Git repositories. It provides a number of additional benefits, including the ability to do the following

- ☐ *Document requirements*
- ☐ *Collaborate on independent streams of history*
- ☐ *Review work in progress*
- ☐ *See team progress*