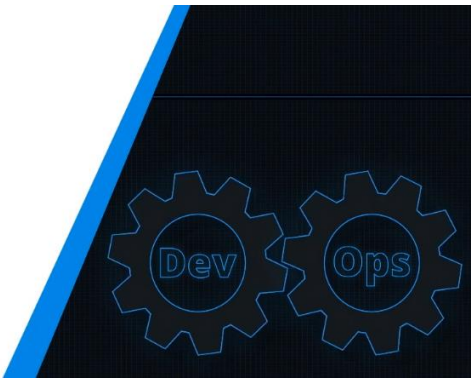
3.1 Learning Objectives

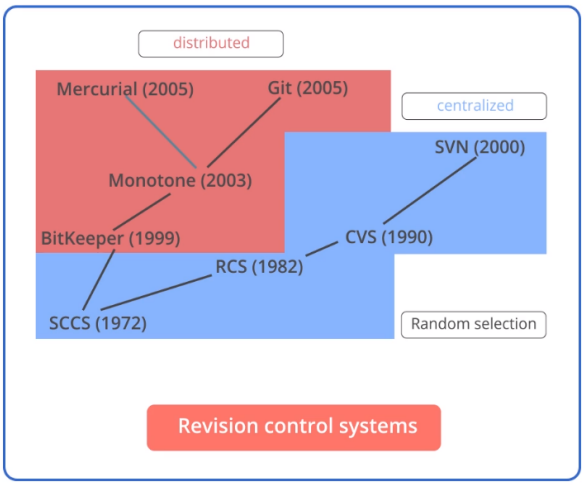
By the end of this lesson, you will be able to:

- Select the suitable version control systems for your organization
- Explain the role of version control systems
- Identify the types of control systems and the supporting tools
- Set up Git
- Identify the suitable source code and version control hosts



3.2 Overview of Version Control Systems

History of Version Control Systems:



A major focus of a version control system (also known as revision control or source control) is to manage the changes to the files, programs, logs, and other information related to code development, code deployment, and code operation.

Need for Version Control Systems:

Files are checked into version control by registered users only

The username and task of the check-in are associated with the revision

Each file check-in gets a new version which is usually a number

All the previous versions of a file can be easily extracted

The latest versions of all files are often referred to as heads

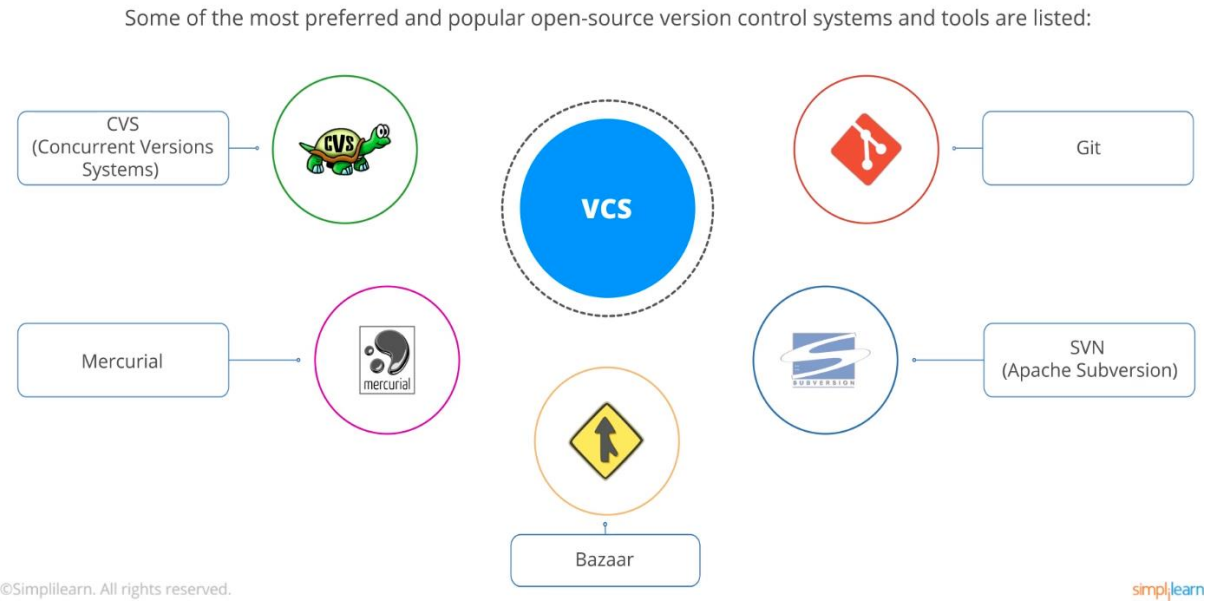
Source control systems are organized into repositories

Version control supports branching where the head of the repository is split for parallel development

Repositories: Dos and Don‘ts:

- Source control management system (SCM) supports multiple projects
- Each project is separated in the SCM system
- Each section is called repository

Popular Version Control Systems:



3.3 Role of Version Control Systems

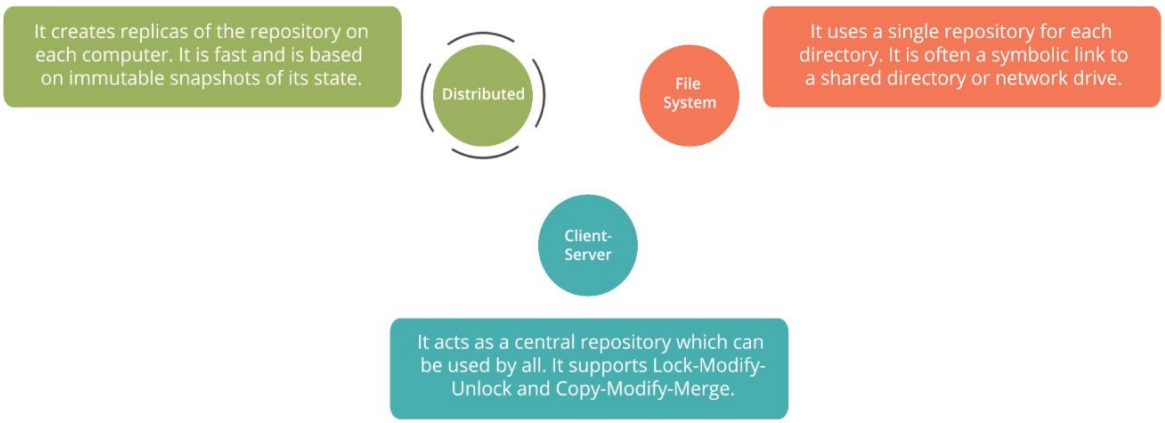
Importance of Version Control Systems:

1. Helps in tracing the development of the documents
2. All the documents can be identified by filenames, authors, and the modified dates
3. Helps in differentiating the latest versions and tracking the recent updates
4. The "truth" files and documents can be stored in a given storage location
5. Helps in keeping track of the independent code versions for various reasons
6. Continuous integration is easier with the newer version control systems
7. Helps in coding collaboratively and remotely

Benefits of Version Control Systems in a DevOps Environment:

3.4 Types of Control Systems and Their Supporting Tools

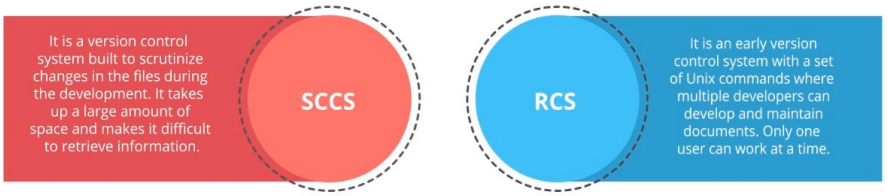
Types of Source Control Systems:



File-based Source Control Systems:

Early source control systems worked at the level of the local file system. It required a subdirectory called SCCS or RCS containing the changes.

Examples of file-based source control systems:



Client-Server-based Source Control Systems:

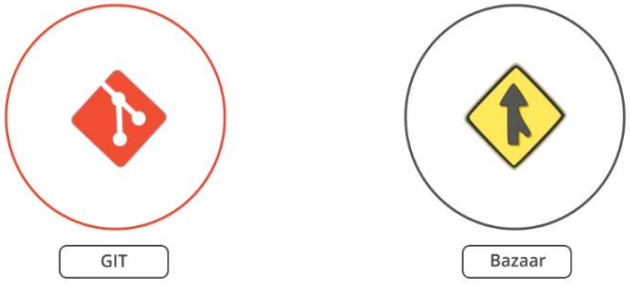
A central repository can be used by all the developers and others who are involved in the product development and maintenance. An early approach to client-server model was Concurrent Versions System (CVS).

Conflicts occur if two people change a file at the same time. This problem can be solved in two ways:

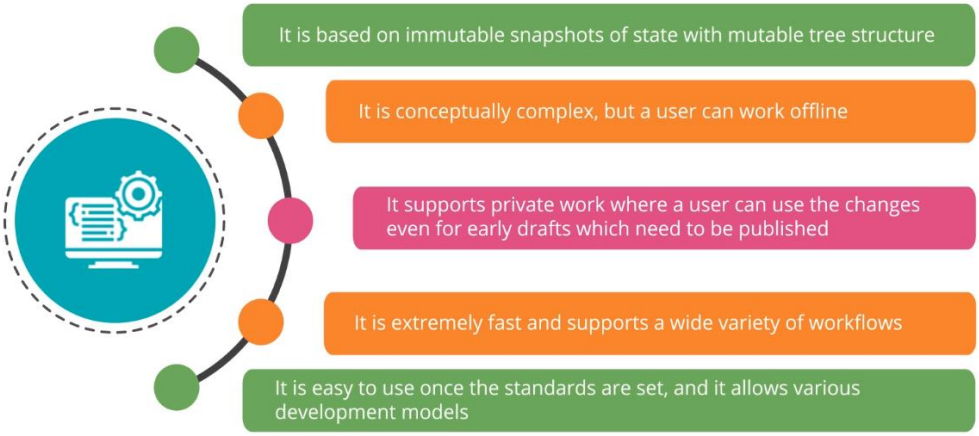
Lock-Modify-Unlock	Copy-Modify-Merge
• Only one person can change a file at a time • A file must be locked before changing it • Different users can't lock a locked file • Another user must wait for the lock to be released	• Each user has a personal working copy of the file tree • Changes are made independently and simultaneously • Private copies are merged into a new version • All other copies become outdated as soon as a private copy is committed

Distributed Source Control Systems:

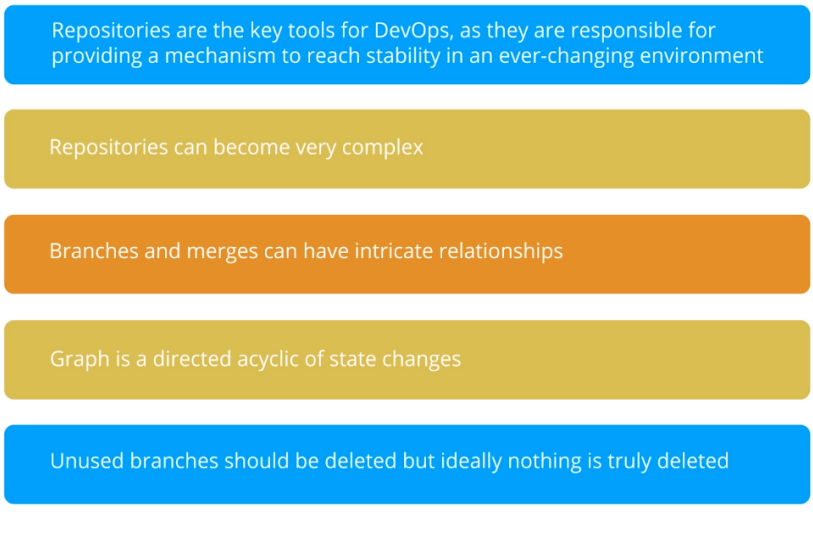
Distributed source control systems create replicas of the repository on each computer. Every user has to work on a replica locally and can do so even being disconnected from the network. They are suited for large projects and independent developers who can work independently and commit the changes for merging.

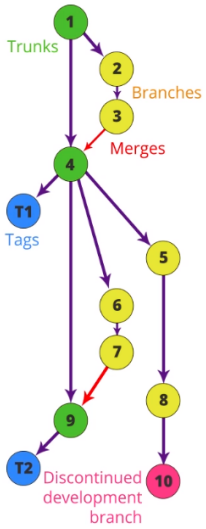


Features of Distributed System:



Repositories are key tools for DevOps:

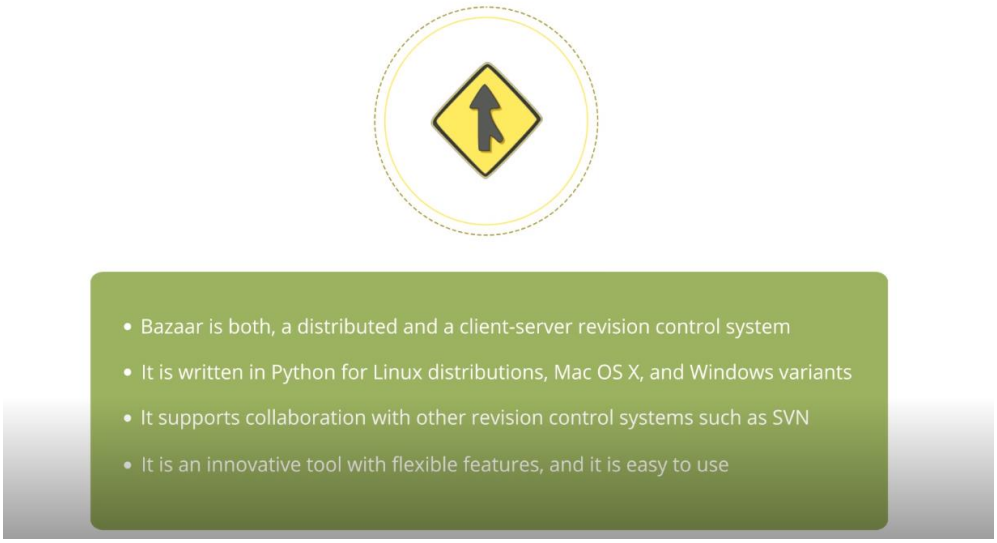




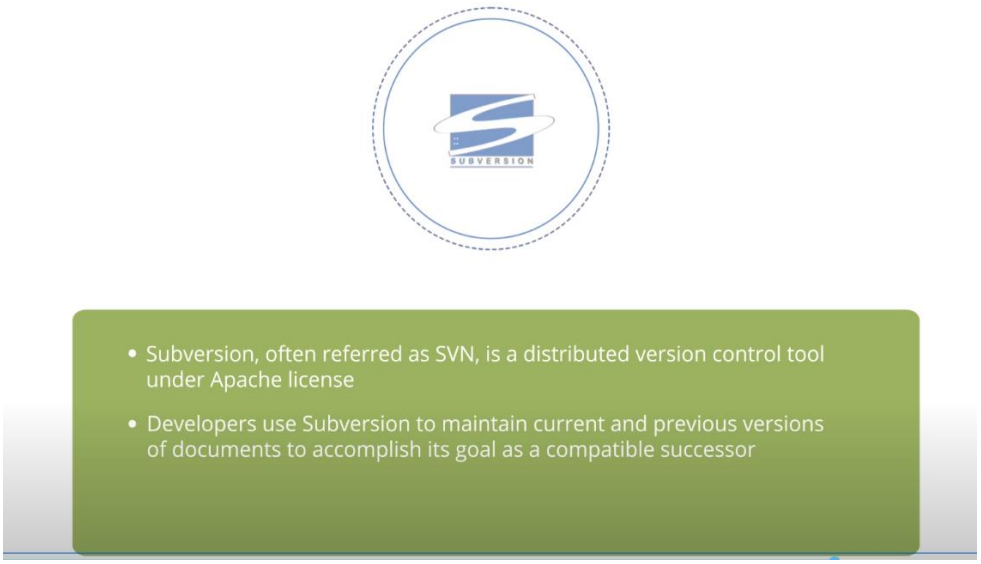
Types of Version Control Systems (Git):

- Git is distributed revision control used to scrutinize the changes
- It is fast, aims for data integrity, and supports nonlinear works
- It supports fast branching and merging
- Branches are lightweight and are able to perform a commit
- Repositories can be published via several protocols such as HTTP, rsync, and FTP

Types of Version Control Systems (Bazaar):



Types of Version Control Systems (Subversion):



Types of Version Control Systems (mercurial):

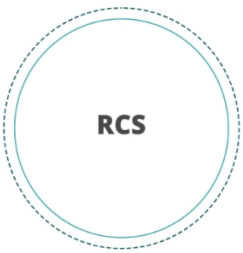
Types of Version Control Systems (cvs):

- CVS (Concurrent Versions System) is a free client-server revision control system which maintains track of all the work and its changes in a set of documents or files and allows to collaborate
- To work well with large text files, it uses delta compression

Types of Version Control Systems (perforce):

- Perforce, referred as Perforce Software, is used for web-based repository management, team collaboration, and agile planning
- It is accepted by larger organizations for better performance on large repositories compared to Subversion
- Perforce is the right choice if an organization has a large source tree and needs a centralized control system

Types of Version Control Systems (rcs):

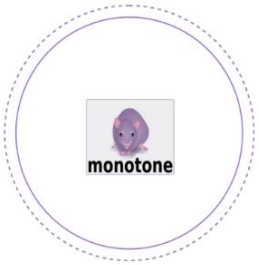


- Revision Control System is an early version control system which helps users in creating their own revisions of a document, commit changes, and merge
- It operates only on a single file and does not support working with an entire project. Though it is simple to work with, there is minimal security offered

Types of Version Control Systems (sccs):

- Source Code Control System is a version control system developed to track the status and the changes of the source code and other documents during the development phase. An SCCS file is inclusive of delta table, access, and tracking tags within a body of text. Some of the popular commands are create, edit, delget, get, and prt

Types of Version Control Systems (monotone):



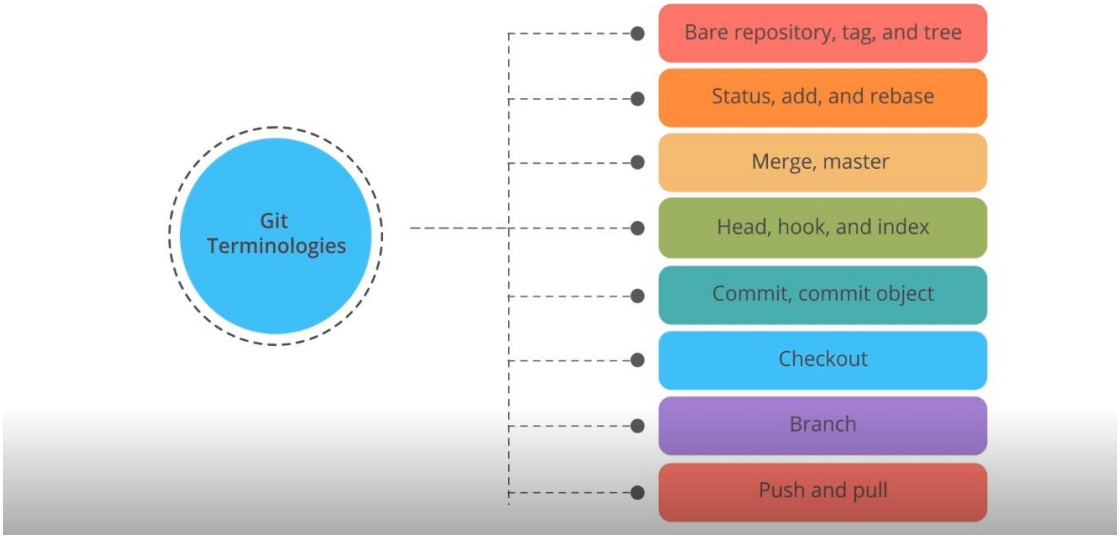
- Monotone is an open-source software tool for distributed revision control
- It helps in tracking revision of codes, documents and also tracks the history across renames
- It strongly supports a diverge/merge workflow
- Its noteworthy features are good support for internationalization and localization, very low maintenance, and stable GUI
- It does not support potential users to checkout from a proxy network due non-HTTP protocol

Types of Version Control Systems (plastic scm):

- Plastic SCM is a cross-platform commercial software tool for distributed version control
- It is a full version control stack which includes a command-line tool, a GUI, different merge tools, and a large number of IDEs
- Its popular features are built-in 3-way merge, directory versioning, distributed and centralized operations

3.5 Overview of Git

Popular Terminologies:



Bare repository, tag and tree:

- **git init–bare** is executed.
- A repository is empty and contains no working or checked out copies of source code.
- A bare repository saves the Git revision history of your repository in the root folder instead of in a .git subfolder. It is customarily given a .git extension.

- A **git tag** is an object that describes a point in history of the version control for a particular repository. Tags are arbitrary.
- A **git tree** describes the hierarchical structure of files in a repository.

Status, add and rebase:

- **git status** reports the state of the local repository on the developer's computer.
- **git add** is a Git command that adds a file from the local workspace into the staging area in preparation for a commit to the local repository.
- **git rebase** is a Git command that will combine a code in one branch with a code from another branch.

Head, hook and index:

- A **git head** indicates the current branch you are working on in the local workspace.
- A **git hook** is an event in which you can program the behavior. Example: Running a series of unit tests against the code before committing it to the local repository. The git will execute the script automatically before a commit takes place. Hooks allow you to extend Git to make your development activity faster and safer.
- The **git index** is the way in which Git keeps track of the state of the local repository. The git index can be considered as the staging area where files are added that are about to be committed.

Commit, commit object:

- A **git commit** is used to add code or a Git object, such as a tree from the local workspace to the local repository.

Checkout:

- A **git checkout** is the Git command that moves code out of the local repository on a developer's computer into the workspace area of the computer's files system.

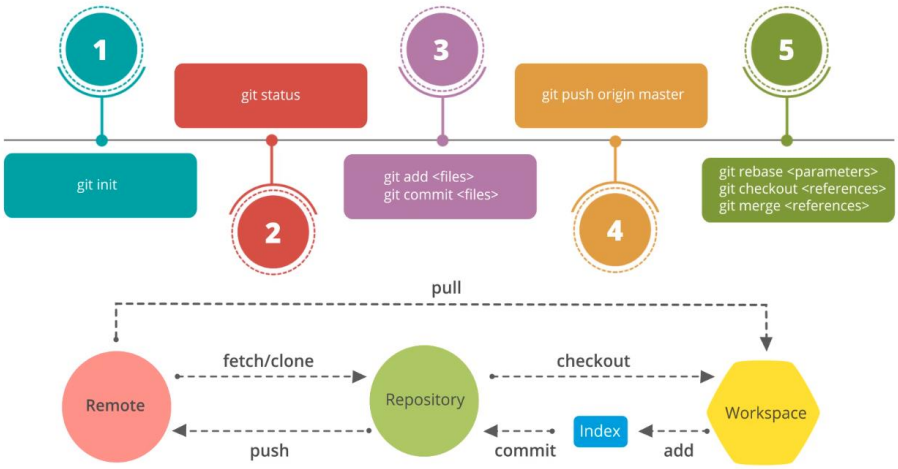
Branch:

- A **git branch** is the Git command that creates a branch in a repository.
- A branch is a copy of another branch that becomes independent once it is created. Thus, a developer has an exact copy of code that can be enhanced safely without affecting the work of others.
- Once work on a branch is complete, that branch can be merged back into the original branch from which it was derived.

Push and pull:

- A **git pull** is the Git command that is used to get the latest code from a Git repository on a remote server to the developer's local machine.
- A **git push** is the Git command that is used to add code from a developer's local repository to a remote repository on the internet or on the company's network.

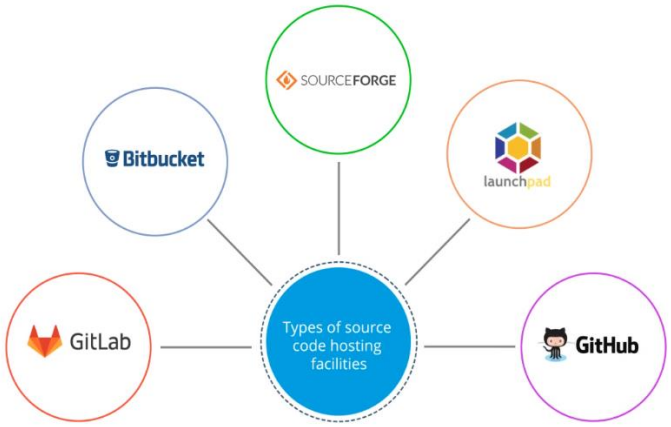
Git Workflow:



3.6. Overview of Source code and Version Control Hosts

Source Code Hosting Facility:

A source code repository is a web hosting and file archive facility where larger data such as source code is stored either privately or publicly. The most popular source code hosting sites are listed:



Supported features:

Name	Code Review	Bug Monitoring	Web Hosting	Build System
GitLab	Yes	Yes	Yes	Yes
Bitbucket	Yes	Yes	Yes	Yes
SourceForge	Yes	Yes	Yes	No
launchpad	Yes	Yes	Yes	Yes (Only Ubuntu)
GitHub	Yes	Yes	Yes	Via third party

Supported Version Control Systems:

Name	CVS	Git	SVN	Perforce
GitLab	No	Yes	No	No
Bitbucket	No	Yes	No	No
SourceForge	Dropped	Yes	Yes	-
launchpad	Import only	Yes	Import only	No
GitHub	No	Yes	Parital	No

Popularity and the Community Usage:

Name	Users	Projects
GitLab	100,000	546,000
Bitbucket	5,000,000	Private
SourceForge	3,700,000	500,000
launchpad	3,965,288	40,881
GitHub	24,000,000	69,000,000

3.7 Deploy the files to GitHub via Git

Exercise

3.9 Key Takeaways

You are now able to:

- Decide the suitable version control systems for your organization
- Explain the role of version control systems
- Identify the types of control systems and the supporting tools
- Set up Git
- Identify the suitable source code and version control hosts