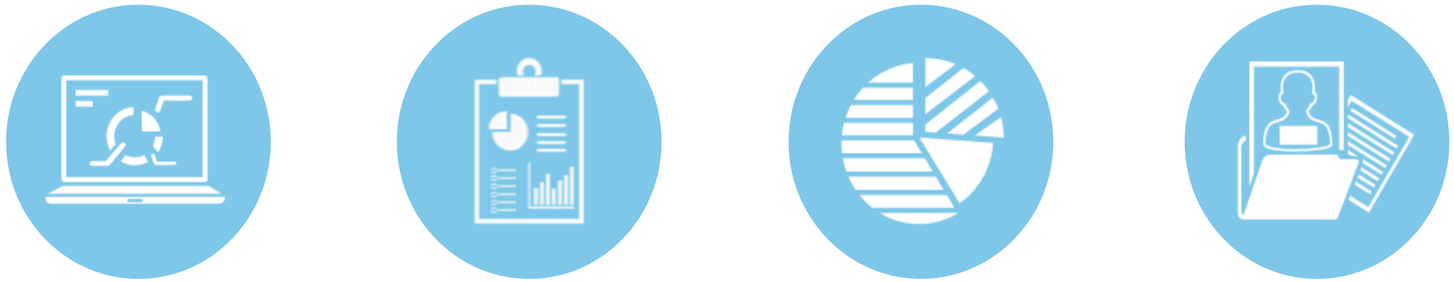
DevOps Certification Training

Lesson 02: Version Control Systems



Learning Objectives

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

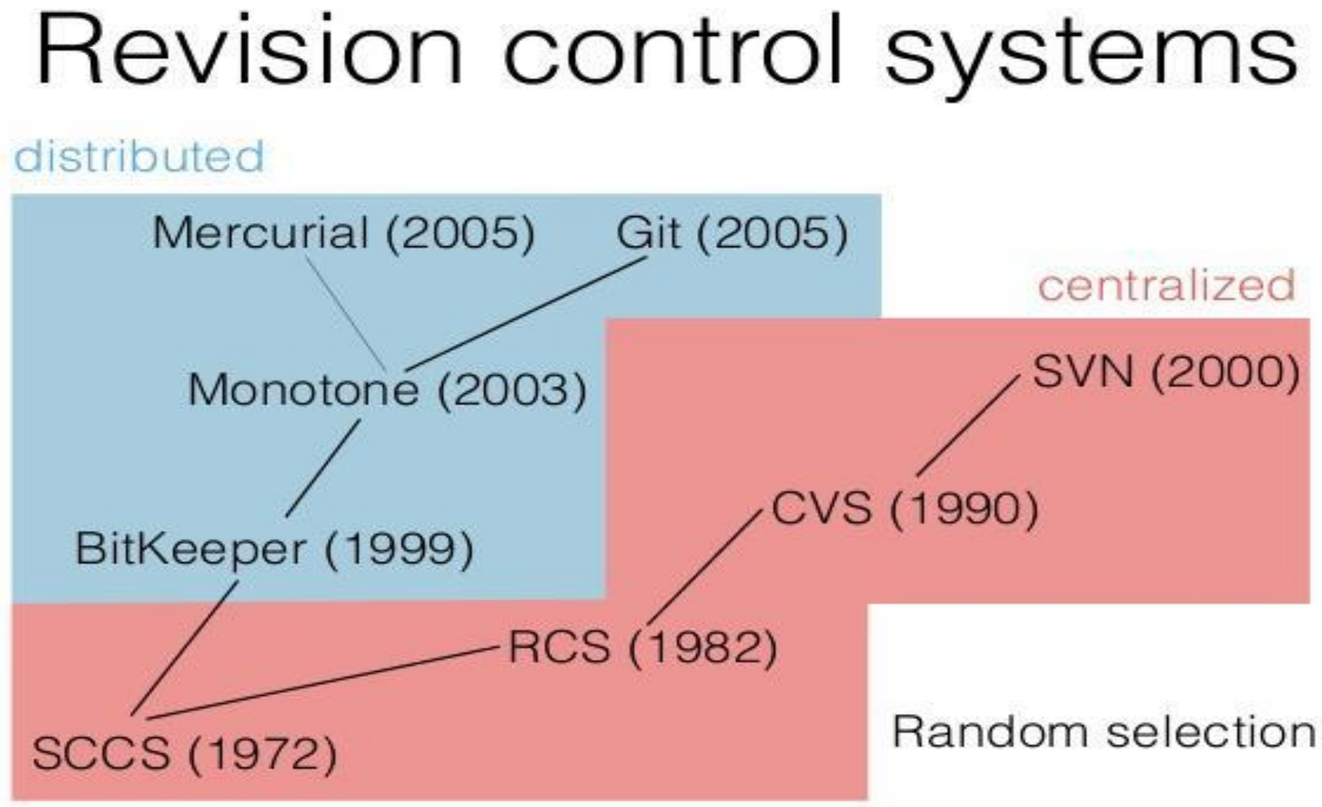
- Select the suitable version control systems for your organization
- Explain the role of version control systems
- Identify the types of control systems and the supporting tools
- Set up Git
- Identify the suitable source code and version control hosts

Version Control Systems

Overview of Version Control Systems

History of Version Control Systems

A major focus of 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 of Version Control System

Version control systems can be used to accomplish various tasks.

Files are checked into version control by registered users only

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

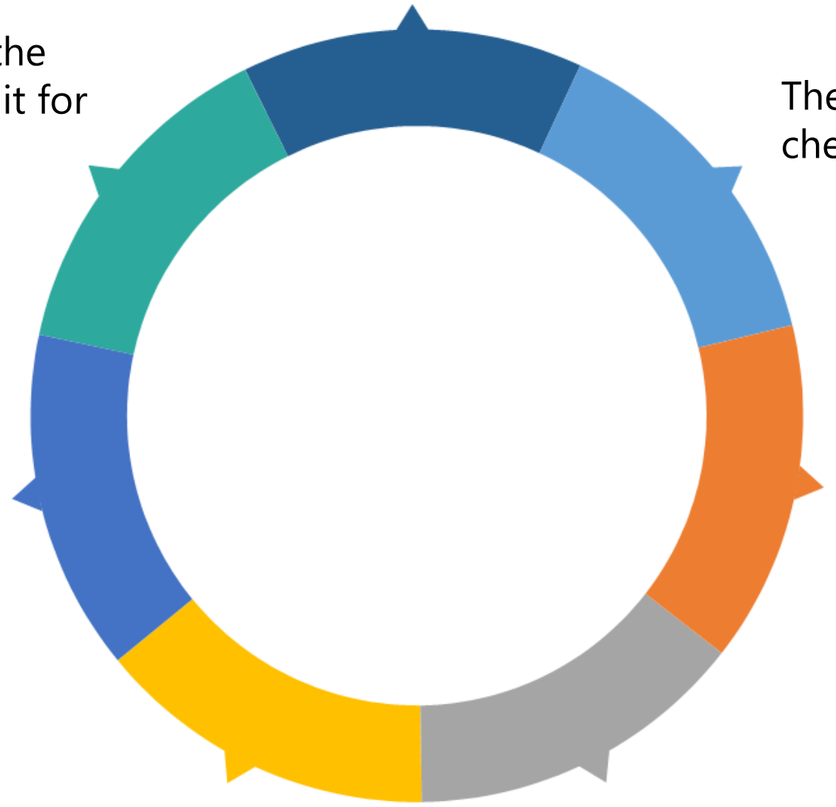
Each file check-in gets a new version which is usually a number

All the previous versions of a file can be easily extracted

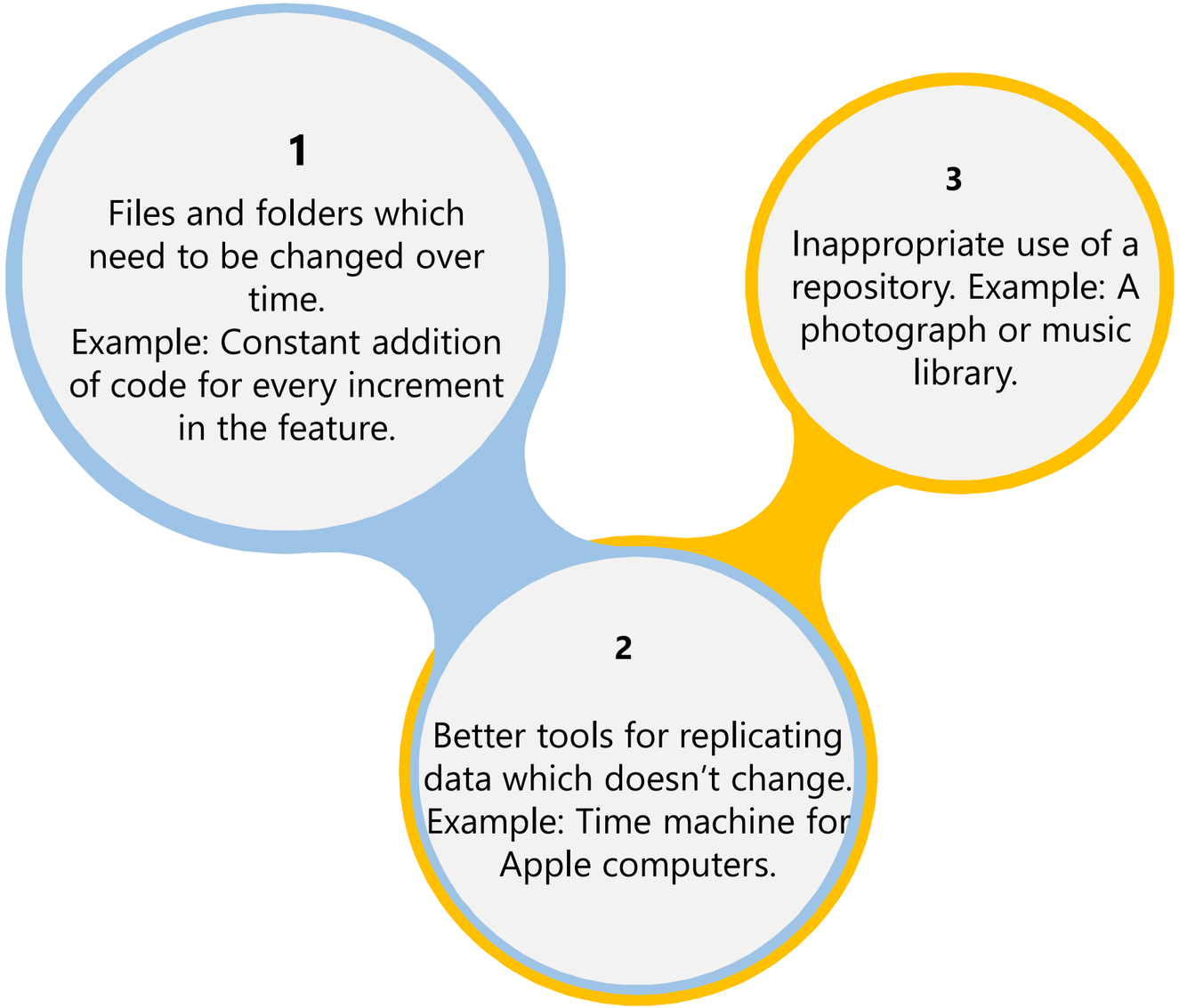
The latest versions of all files are often referred to as heads

Source control systems are organized into repositories

Supports branching where the head of the repository is split for parallel development

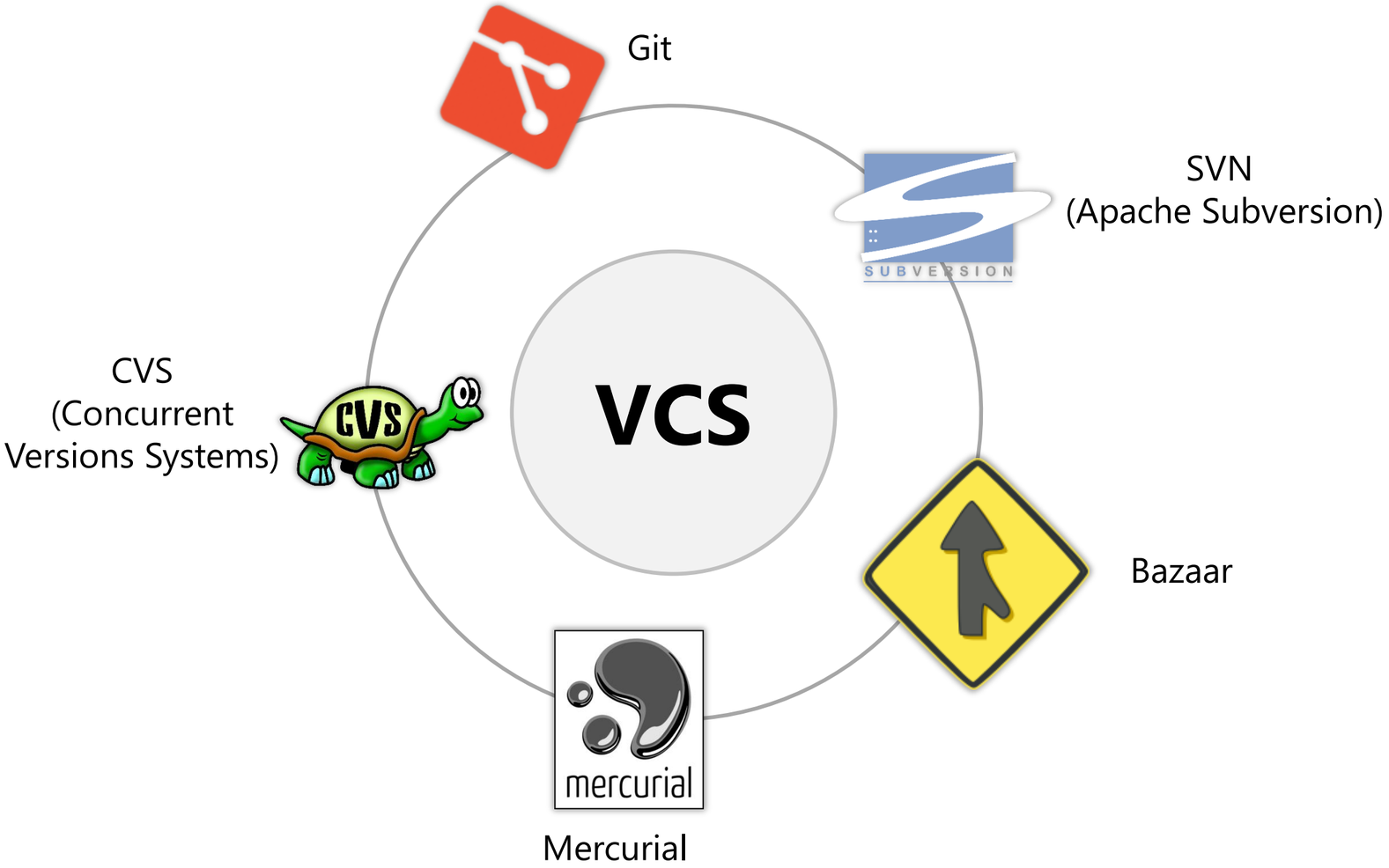


Repository Usage



Popular Version Control Systems

Some of the most preferred and popular open-source version control systems and tools are listed:



Version Control Systems

Role of Version Control Systems

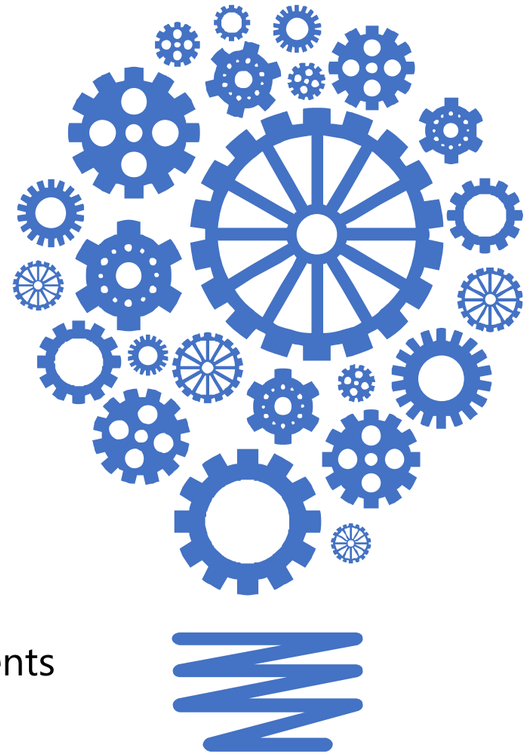
Importance of Version Control System

1. Helps in tracing the development of the documents such as policies or source codes
2. All the documents can be identified by filenames, authors, and the modified dates. It helps in linking contributions and contributors
3. Helps in differentiating the latest versions and track 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

Role of Version Control System in DevOps Environment

Organize product-related artifacts

Adopt peer reviews to inspect changes



Provide tried and tested guaranteed results

Regulate monitoring of documents

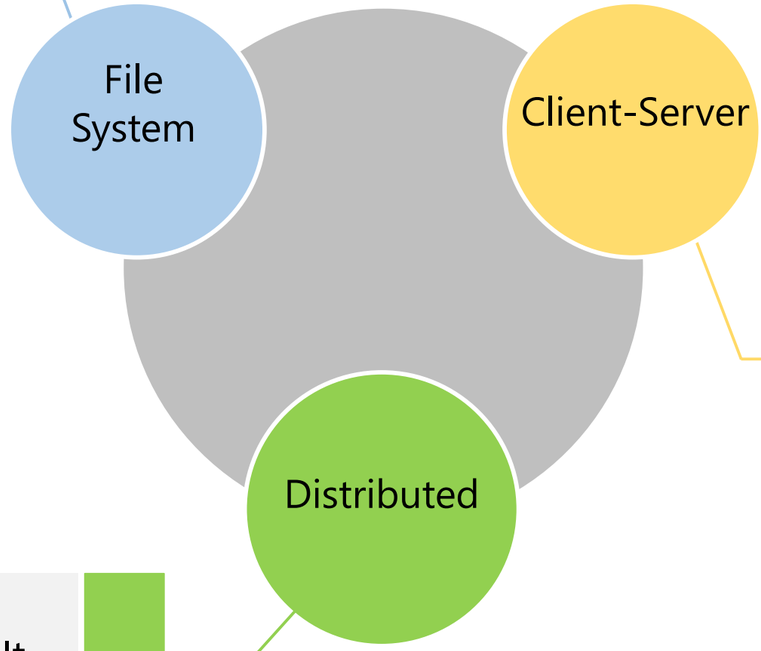
Establish win-win relationship between Dev and Ops

Version Control Systems

Types of Control Systems and Its Supporting Tools

Types of Source Control Systems

It uses a single repository for each directory. It is often a symbolic link to a shared directory or network drive.



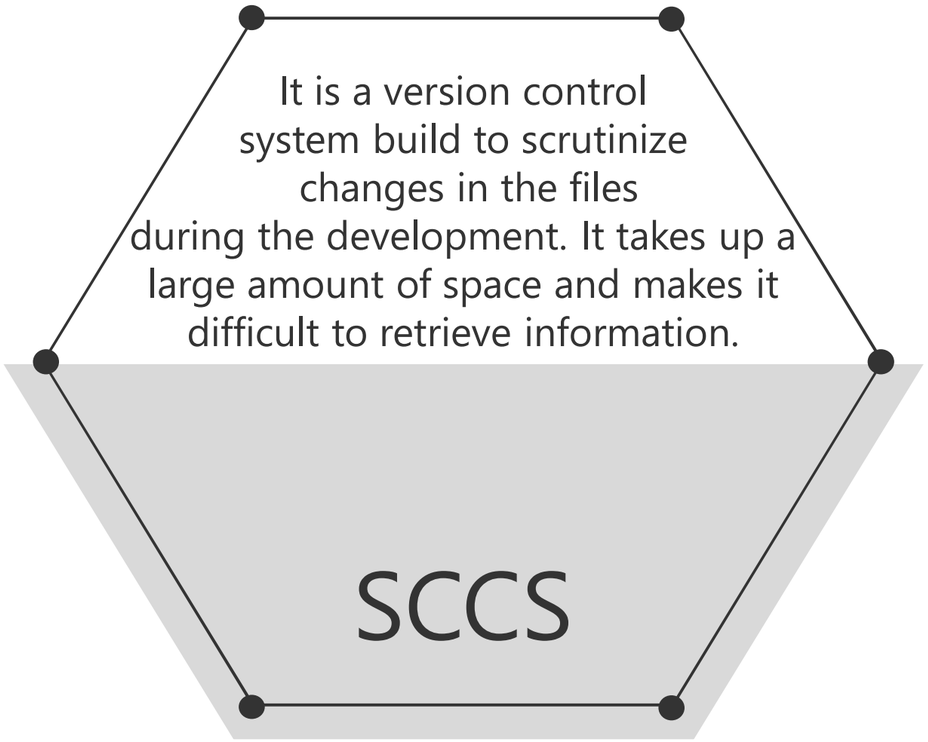
It acts as a central repository which can be used by all. It supports Lock-Modify-Unlock and Copy-Modify-Merge.

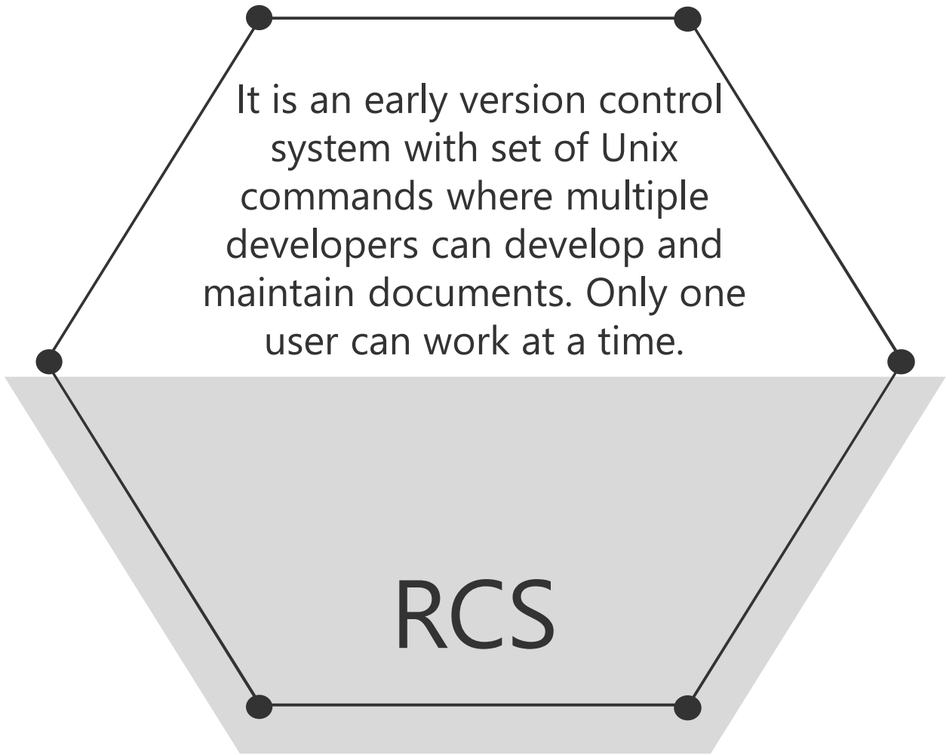
It creates replicas of the repository on each computer. It is fast and is based on immutable snapshots of its state.

File-Based Source Control System

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 System

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

- 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

Copy-Modify-Merge

- 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 System

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.

Examples:

GIT

Bazaar

Features of Distributed System

It is based on immutable snapshots of state with mutable tree structure

It is conceptually complex, but a user can work offline

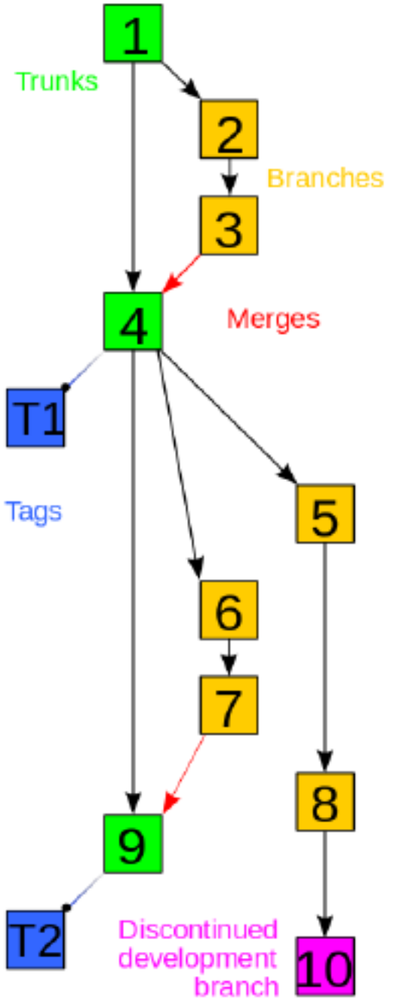
It supports private work where a user can use the changes even for early drafts which need to be published

It is extremely fast and supports a wide variety of workflows

It is easy to use once the standards are set, and it allows various development models.

Repositories Are Key Tools in DevOps

- Repositories are key tools for DevOps, as they are responsible for providing a mechanism to reach stability in an ever-changing environment.
- Repositories can become very complex
- Branches and merges can have intricate relationships
- Graph is a directed acyclic of state changes
- Unused branches should be deleted but ideally nothing is truly deleted.



Types of Version Control Systems

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

Bazaar is both, a distributed and a client-server revision control system written in Python for Linux distributions, Mac OS X and Windows variants. It supports collaboration with other revision control Systems such as SVN. It is an innovative tool with flexible features, and it is easier to use.

Types of Version Control Systems

Subversion, often referred as SVN is a distributed version control tool under Apache license. Developers use subversion to maintain current and previous versions of documents to accomplish its goal as a compatible successor. Path and Revision are the two coordinates which are used to address file system.

Mercurial is a distributed revision control tool which supports Microsoft Windows and Unix-like environments. It is highly scalable, decentralized, and efficient. It is easier to learn, but add-ons should be written in Python. It does not support partial checkouts which is a drawback while working on large projects.

Types of Version Control Systems

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.

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 user 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.

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 and body of the text. Some of the popular commands are create, edit, delget, get, and prt.

Tools Supporting Version Control Systems

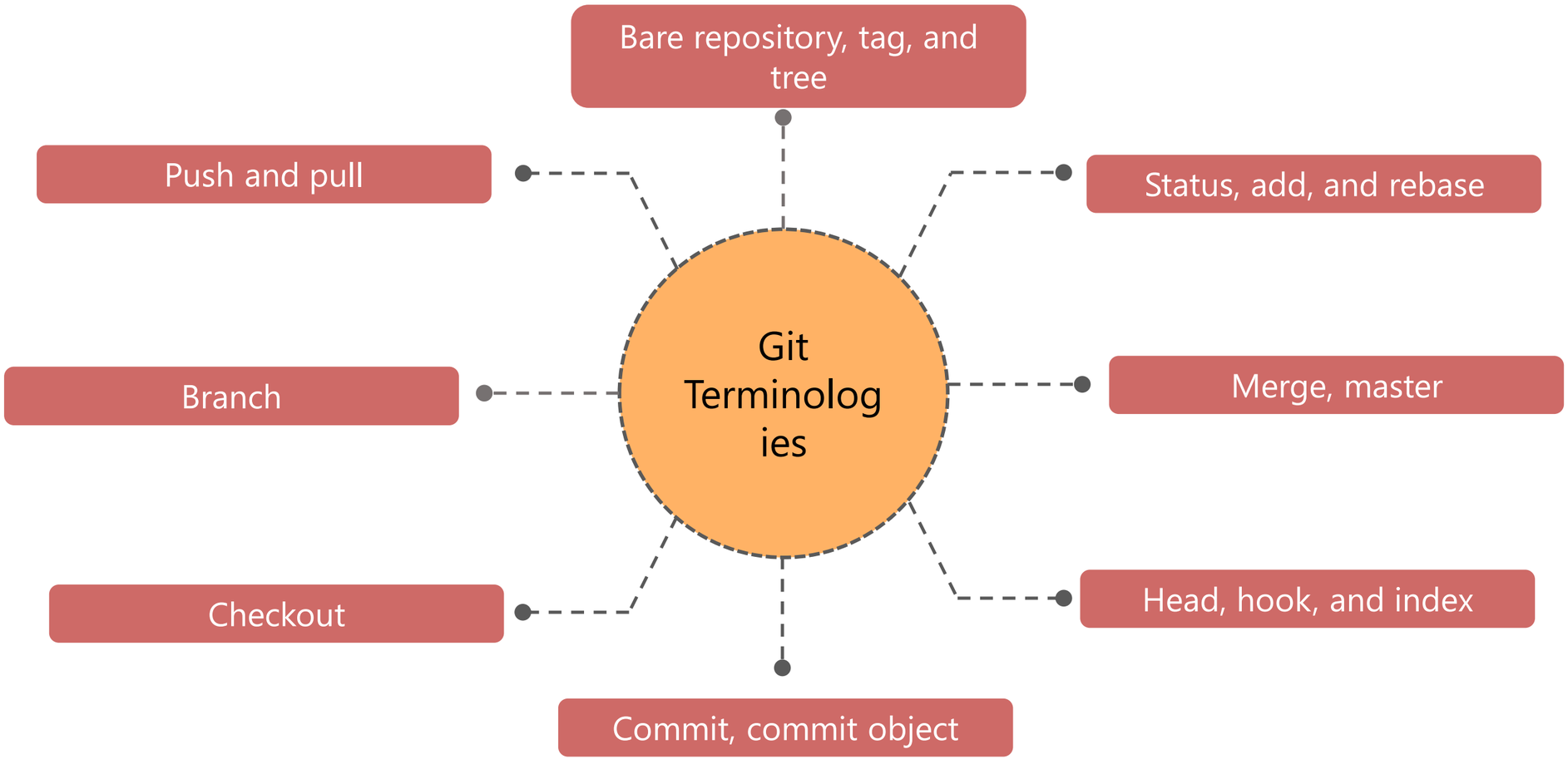
Monotone is an open-source software tool for distributed revision control. It helps in tracking revision of codes, documents and also tracks the history across renames. It strongly supports a diverge/merge workflow. Its noteworthy features are good support for internationalization and localization, very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

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. It supports popular operating systems such as Mac OS X, Windows, and few Linux platforms.

Version Control Systems

Overview of Git

Popular Terminologies



Git Workflow

1

git init

2

git status

3

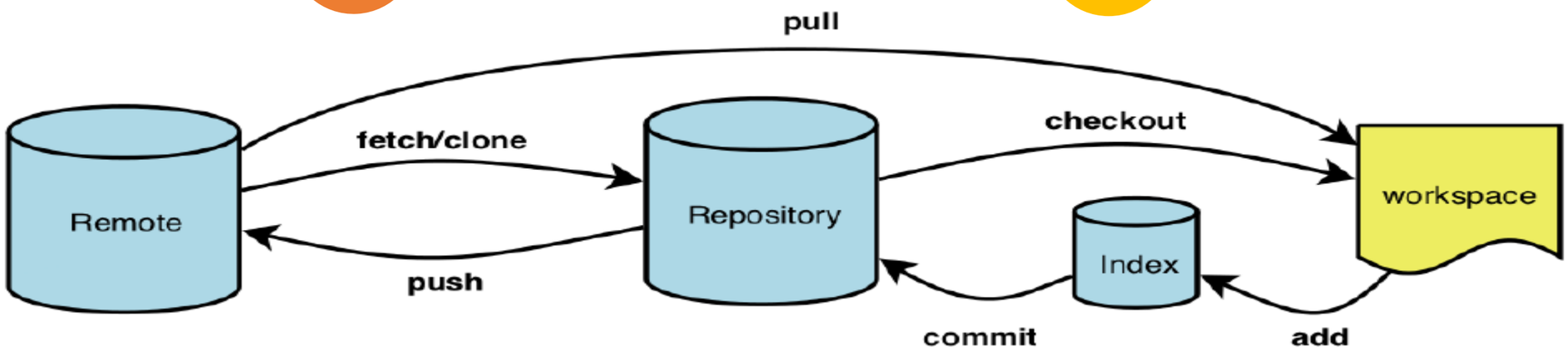
git add <files>
git commit <files>

4

git push origin master

5

git rebase <parameters>
git checkout <references>
git merge <references>

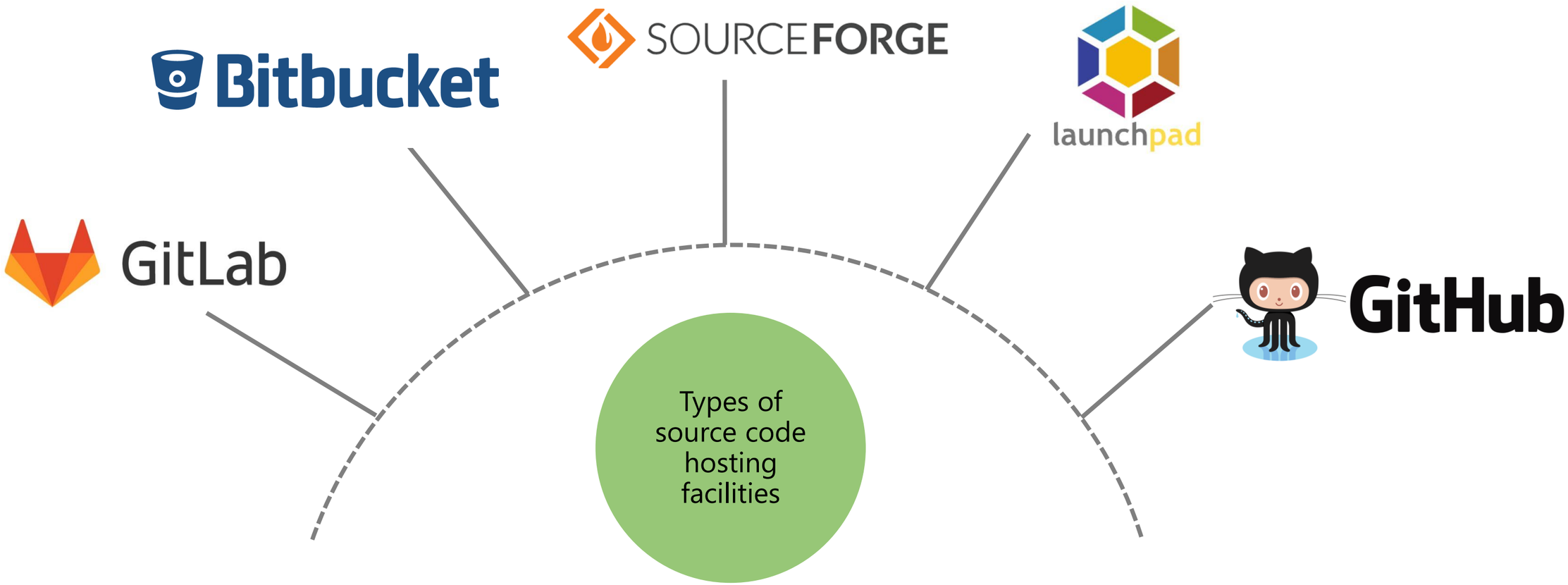


Version Control Systems

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 Tracking	Web Hosting	Build System
GitLab	Yes	Yes	Yes	Yes
Bitbucket	Yes	Yes	Yes	Yes
SourceForge	Yes	Yes	Yes	No
launchpad	Yes	Yes	No	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	Partial	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

Assisted Practice

Duration: 40 mins

Deploy the Files to GitHub via Git.

Problem Statement: You are given a project to add all the necessary files to your GitHub account via Git interactive terminal.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Assisted Practice: Guidelines to Deploy Files to Hosting Servers

1. List the basic structure of Git, GitHub, and Bitbucket.
2. Create files with every extension, and add random content in it (.txt, .html, .css, .ts, .js, .java, .c).
3. Initialize the Git repo locally and commit the files.
4. Make changes in random files, and repeat step 2.
5. Sign in to the GitHub account. Create an account if it is not created.
6. Connect the local and remote repositories.
7. Add the codes to the repositories of GitHub.
8. Repeat this demonstration three times where each time the commit should be for different repositories.

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

Knowledge Check

Knowledge Check 1	**Choose the software tool that supports Version Control System.**

a. Perforce

b. Plastic SCM

c. Git

d. CVS

Knowledge Check

1

Choose the software tool that supports Version Control System.

a. Perforce

b. Plastic SCM

c. Git

d. CVS

The correct answer is **b. Plastic SCM**

Knowledge Check

2

Choose the correct File-based version control system.

a. SCCS and RCS

b. Perforce and SVN

c. Git and SCCS

d. SCCS , RCS, and Git

Knowledge Check

2

Choose the correct File-based version control system.

a. SCCS and RCS

b. Perforce and SVN

c. Git and SCCS

d. SCCS , RCS, and Git

The correct answer is **a. SCCS and RCS**

Knowledge Check

3

Choose the correct solutions for the conflict raised in client-server based systems.

a. Lock-Modify-Unlock

b. Copy-Modify-Merge

c. Create-Modify-Merge

d. Options a and b

Knowledge Check

3

Choose the correct solutions for the conflict raised in client-server based systems.

a. Lock-Modify-Unlock

b. Copy-Modify-Merge

c. Create-Modify-Merge

d. Options a and b

The correct answer is **d. Options a and b**

Knowledge Check

4

A built-in 3-way merge is a popular feature of _____

a. Bazaar

b. Plastic SCM

c. Perforce

d. Git

Knowledge Check

4

A built-in 3-way merge is a popular feature of _____

a. Bazaar

b. Plastic SCM

c. Perforce

d. Git

The correct answer is **b. Plastic SCM**

Knowledge Check

5

Which among the following does not support the build system?

a. SourceForge

b. GitLab

c. GitHub

d. None of the above

Knowledge Check

5

Which among the following does not support the build system?

a. SourceForge

b. GitLab

c. GitHub

d. None of the above

The correct answer is **a. SourceForge**

Lesson-End Project

Deploy the Files to Bitbucket via Git.

Duration: 40 mins

Problem Statement:

Create a directory named “Lesson-02” and perform the following:

- Create multiple files with .html, .xml, .txt, .py, .js, and .java.
- Add random content in each file.
- Connect git local to Bitbucket remote.
- Push the directory “Lesson-02” to Bitbucket account.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You