Objective:

After studying this experiment you will be able to:

- Understand the concept of software configuration management.
- Use of automated tools for software configuration management process.

Outcome:

- Able to understand how to modify system under use, to make it capable to adopt current changes and upgrade it.
- Ability to use readymade tools for adopting, controlling and versioning proper changes for upgrading the system under use.

Software Used: SCM tool- Git-GitHub

Theory: Software Configuration Management using open source Tool Git-GitHub

SCM Tool Steps

- **1) Install Git**
    1. Browse to the official Git website: https://git-scm.com/downloads
    2. Click the download link for Windows and allow the download to complete.
    3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.
    4. Allow the app to make changes to your device by clicking Yes on the User Account Control dialog that opens.
    5. Review the GNU General Public License, and when you're ready to install, click Next.
    6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click Next.
    7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click Next.
    8. The installer will offer to create a start menu folder. Simply click Next.
    9. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click Next.
    10. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click Next.
    11. This installation step allows you to change the PATH environment. The PATH

is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click Next.

12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click Next.

The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click Next.

13. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click Next.
14. Choose the terminal emulator you want to use. The default Min TTY is recommended, for its features. Click Next.
15. The installer now asks what the `git pull` command should do. The default option is recommended unless you specifically need to change its behavior. Click Next to continue with the installation.
16. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click Next.
17. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click Next.
18. Depending on the version of Git you're installing, it may offer to install experimental features. At the time this article was written, the options to include support for pseudo controls and a built-in file system monitor were offered. Unless you are feeling adventurous, leave them unchecked and click Install.
19. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click Finish

- **2) Start Git**

Git has two modes of use – a bash scripting shell (or command line) and a graphical user interface (GUI).

1. To launch Git Bash open the Windows Start menu, type *git bash* and press Enter (or

click the application icon).

2. To launch Git GUI open the Windows Start menu, type *git gui* and press Enter (or click the application icon).

Create a Test Directory

Open a Windows PowerShell interface by pressing Windows Key + x, and then i once the menu appear.

Create a new test directory (folder) by entering the following:

```
mkdir git_test
```

Change your location to the newly created directory:

```
cd git_test
```

Configure GitHub Credentials

Configure your local Git installation to use your GitHub credentials by entering the following:

```
git config --global user.name "github_username"
```

```
git config --global user.email "email_address"
```

Clone a GitHub Repository

Go to your repository on GitHub. In the top right above the list of files, open the Clone or download drop-down menu. Copy the URL for cloning over HTTPS.

Switch to your PowerShell window, and enter the following:

```
git clone repository_url
```

List Remote Repositories

You're working directory should now have a copy of the repository from GitHub. It should contain a directory with the name of the project. Change to the directory:

```
cd git_project
```

Once you're in the sub-directory, list the remote repositories:

```
git remote -v
```

Pushing Local Files to the Remote Repository

Once you've done some work on the project, you may want to submit those changes to the remote project on GitHub.

1. For example, create a new text file by entering the following into your PowerShell window:

```
new-item text.txt
```

2. Now check the status of your new Git branch and untracked files:

```
git status
```

3. Add your new file to the local project:

```
git add text.txt
```

4. Run git status again to make sure the text.txt file has been added. Next, commit the changes to the local project:

```
git commit -m "Sample 1"
```

5. Finally, push the changes to the remote GitHub repository:

```
git push
```

Conclusion: Hence, we have studied software configuration management using open source tool Git-GitHub.