**DEVOPS EXPERIMENT 2**

**AIM:** To understand Version Control System / Source Code Management, install git and create a GitHub account.

**LO:** L2-To obtain complete knowledge of the “version control system” to effectively track changes augmented with Git and GitHub

**What is version control?**

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter. They are especially useful for [DevOps](https://www.atlassian.com/devops/what-is-devops) teams since they help them to reduce development time and increase successful deployments.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members. Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree". One developer on the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree.

Version control helps teams solve these kinds of problems, tracking every individual change by each contributor and helping prevent concurrent work from conflicting. Changes made in one part of the software can be incompatible with those made by another developer working at the same time. This problem should be discovered and solved in an orderly manner without blocking the work of the rest of the team.

**Why use version control?**

Software is developed to solve a user problem. Increasingly, these solutions have many different forms (e.g. mobile, embedded, SaaS) and run a variety of environments, such as cloud, on-perm, or Edge.

As organizations accelerate delivery of their software solutions through DevOps, controlling and managing different versions of application artefacts - from code to configuration and from design to deployment - becomes increasingly difficult. Velocity without robust version control and traceability is like driving a car without a seatbelt.

Version control facilitates coordination, sharing, and collaboration across the entire software development team. Version control software enables teams to work in distributed and asynchronous environments, manage changes and versions of code and artefacts, and resolve merge conflicts and related anomalies.

**Source code management**

When multiple developers are working within a shared codebase it is a common occurrence to make edits to a shared piece of code. Separate developers may be working on a seemingly isolated feature, however this feature may use a shared code module. Therefore developer 1 working on Feature 1 could make some edits and find out later that Developer 2 working on Feature 2 has conflicting edits.

Before the adoption of SCM this was a nightmare scenario. Developers would edit text files directly and move them around to remote locations using FTP or other protocols. Developer 1 would make edits and Developer 2 would unknowingly save over Developer 1’s work and wipe out the changes. SCM’s role as a protection mechanism against this specific scenario is known as [Version Control](https://www.atlassian.com/git/tutorials/what-is-version-control).

In addition to version control SCM provides a suite of other helpful features to make collaborative code development a more user friendly experience. Once SCM has started tracking all the changes to a project over time, a detailed historical record of the projects life is created.

**The importance of source code management tools**

Before the adoption of SCM this was a nightmare scenario. Developers would edit text files directly and move them around to remote locations using FTP or other protocols. Developer 1 would make edits and Developer 2 would unknowingly save over Developer 1’s work and wipe out the changes. SCM’s role as a protection mechanism against this specific scenario is known as [Version Control](https://www.atlassian.com/git/tutorials/what-is-version-control).

SCM brought version control safeguards to prevent loss of work due to conflict overwriting. These safeguards work by tracking changes from each individual developer and identifying areas of conflict and preventing overwrites. SCM will then communicate these points of conflict back to the developers so that they can safely review and address.

## **Steps For Installing Git for Windows**

Installing Git prompts you to select a text editor. If you don’t have one, we strongly advise you to install prior to installing Git. Our roundup of the [best text editors for coding](https://phoenixnap.com/kb/best-linux-text-editors-for-coding) may help you decide.

### Download Git for Windows

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.

### Extract and Launch Git Installer

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**.

### Server Certificates, Line Endings and Terminal Emulators

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.**

13. 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**.

14. 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**.

15. Choose the terminal emulator you want to use. The default Minty is recommended, for its features. Click **Next**.

16. The installer now asks what the git pull command should do. The default option is recommended unless you specifically need to change its behaviour. Click **Next**to continue with the installation.

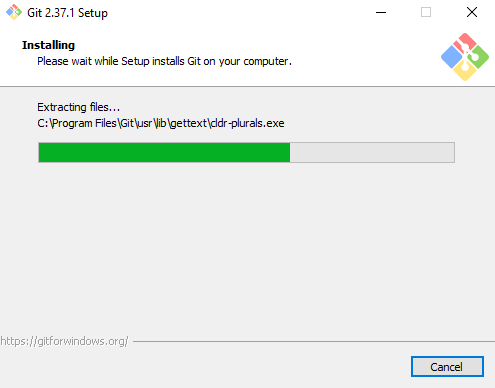
17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **next**.

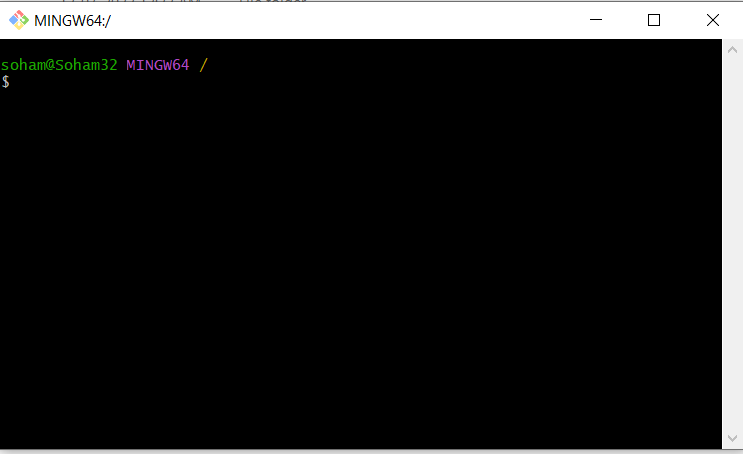
18. The default options are recommended, however this step allows you to decide which extra option you would like to enable. If you use symbolic links, which are like shortcuts for the command line, tick the box. Click **Next**.

19. 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**.

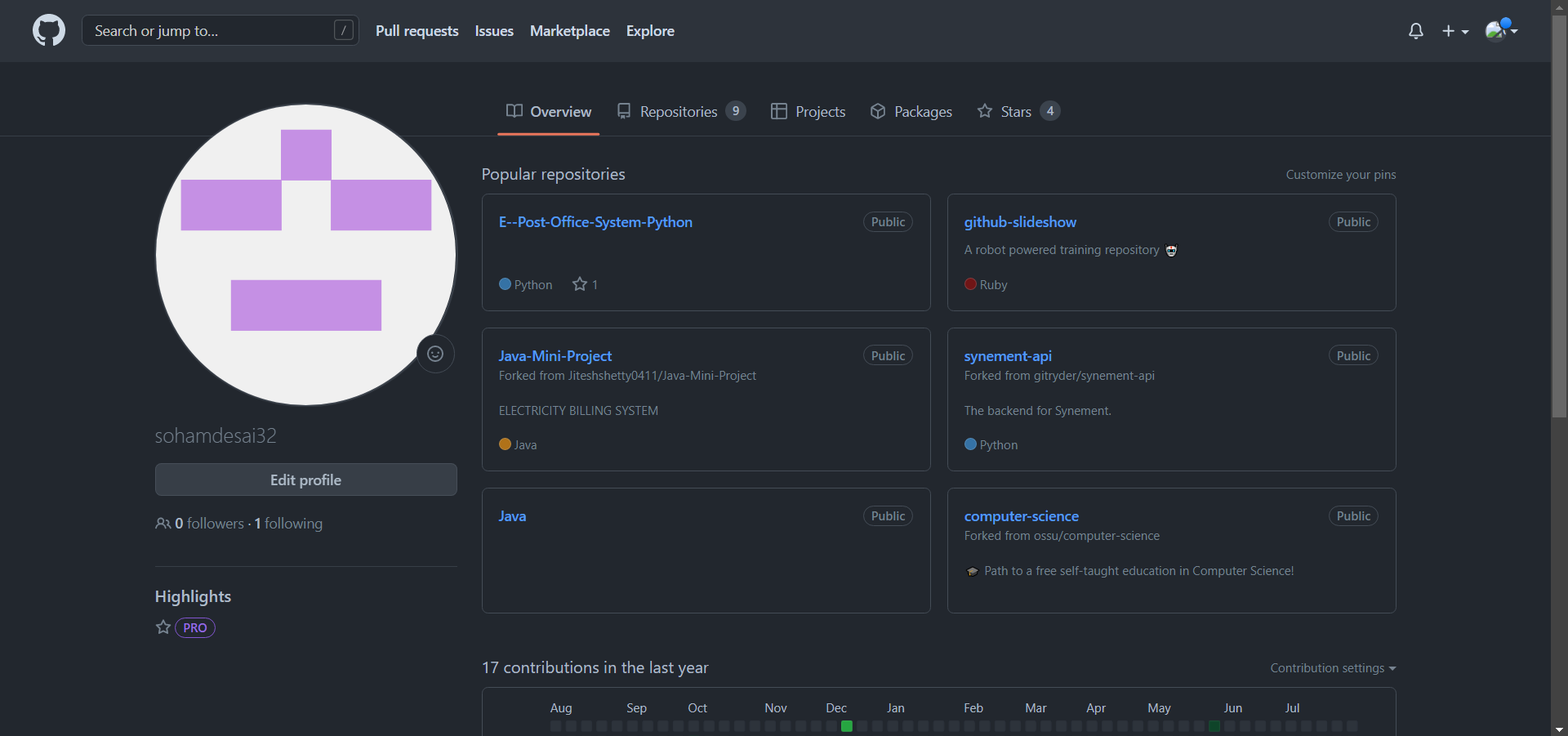
20. Once the installation is complete, tick the boxes to view the Release Notes or Launch Git Bash, then click **Finish**.

**DOWNLOAD GIT:**



****

**CREATE A GITHUB ACCOUNT:**

****

**CONCLUSION:**

In this experiment we learned about git as a version control system which is used in keeping track of the files in a particular file directory and we can always review the changes at any point in time and we also learned about GitHub which is essentially a hosting website for our code.