Experiment 02 - Version Control

Roll No.	24
Name	Iyer Sreekesh Subramanian
Class	D15-A
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements LO2: To obtain complete knowledge of the “version control system” to effectively track changes augmented with Git and GitHub

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

Introduction:

(Introduction to Version Control and its importance)

Version Control is a method to record changes to a file or set of files over time so that you can recall specific versions later. In the case of software, you will use your software source code as the files being version controlled, though, in reality, you can do this with nearly any type of file on a computer.

The systems used to track changes and different code versions are called Version Control Systems (VCS). Just like with anything else, each VCS has its unique features and comes with its own set of advantages and disadvantages. There are some basics, though, that make a VCS what it is.

They retain a long-term history of changes made on a project including creation, deletion, edits and more. This should include the author, date and any notes from the changes as well.

Beyond that, they should have a solution for branching and merging new code changes to the main project to allow concurrent work from multiple members of a team. That is the point, after all. The main codebase for a project is called the Trunk or, simply, Main. Branches are created as independent streams of work that can be merged to the Main. All of this supports traceability in projects. If something goes wrong, it’s easier to look back at the changes made over time and connect them to bugs and errors when they appear.

Importance of Version Control in DevOps:

1. **Avoiding Dependency issues in modern applications:**
   Dependency issues are common in applications across multiple languages and have been since early programming days. Version Control plays a huge role here. Access to the different code iterations can reveal where new changes expose dependencies that clash with other parts of the code.

2. **Providing better performance:**
   VCS can view and understand how changes to one part of the code cause problems across the application. It enables coding practices like Continuous Integration and, as a result, Continuous Delivery/Deployment.

3. **Improving Application Reliability**
   VCS can handle smaller changes more frequently. It can also track those changes so that everyone is looking at the same thing and so that troubleshooting is easier in the case of failures.

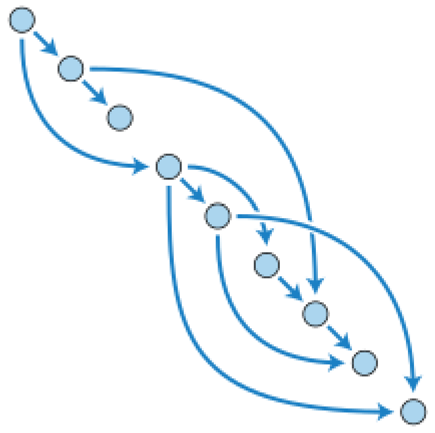
<u>**Git**</u>:

(Explain Git architecture and terminologies used)

Git is a distributed version control system (DVCS) for tracking changes to files. But what does that mean? Git is an open-source VCS, which is not file-based, unlike other systems. Rather, it stores information as snapshots. Being a VCS, helps coders revert to their previous code when they hit a roadblock in the newer version, without affecting the source code. On the other hand, what makes it different from other VCS is the way it sees data, which is more like a series of snapshots.

Git Architecture:

Git uses a Directed Acyclic Graph (DAG) for content storage as well as commit and merge histories. A DAG is a finite directed acyclic graph. Being acyclic means that there is no way to go from Node A to Node B and loop back to Node A through any number of edges. A DAG also must have topological ordering. This means that the vertices all have edges that are directed from earlier to later in the sequence.



Git Terminologies:

1. **Repository:** It is a directory that stores all the files, folders, and content needed for your project. It's the object database of the project, storing everything from the files themselves to the versions of those files, commits, deletions, etc.
2. **Branch:** A version of the repository that diverges from the main working project. Branches can be a new version of a repository, experimental changes, or personal forks of a repository for users to alter and test changes.
3. **Fork:** Creating a copy of a repository.
4. **Remote:** Updates a remote branch with the commits made to the current branch. You are literally "pushing" your changes onto the remote.
5. **HEAD:** HEAD is a reference variable used to denote the most current commit of the repository in which you are working. When you add a new commit, HEAD will then become that new commit.
6. **Master:** It is the primary branch of all repositories. All committed and accepted changes should be on the master branch.
7. **Merge:** Taking the changes from one branch and adding them into another (traditionally master) branch. These commits are usually first requested via pull request before being merged by a project maintainer.
8. **Origin:** The conventional name for the primary version of a repository. Git also uses *origin* as a system alias for pushing and fetching data to and from the primary branch. For example, *git push origin master*, when run on a remote, will push the changes to the master branch of the primary repository database.
9. **Pull Requests:** If someone has changed code on a separate branch of a project and wants it to be reviewed to add to the master branch, that someone can put in a pull request. Pull requests ask the repo maintainers to review the commits made, and then, if acceptable, merge the changes upstream. A pull happens when adding the changes to the master branch.
10. **Push:** Updates a remote branch with the commits made to the current branch. You are literally "pushing" your changes onto the remote.
11. **Stash:** While working with Git, you may need to make multiple changes to files, but you may not want all changes to go in one commit. To put temporary changes on hold, you can "stash" your changes, essentially clearing them from the staging area until the changes are called again. You can only stash one set of changes at a time. To stash your staging area use *git stash [files]*; to retrieve the stashed files, run git stash pop. You can also clear the stashed files with *git stash drop*.

Installation:

(Explain steps for installing Git with screenshots)

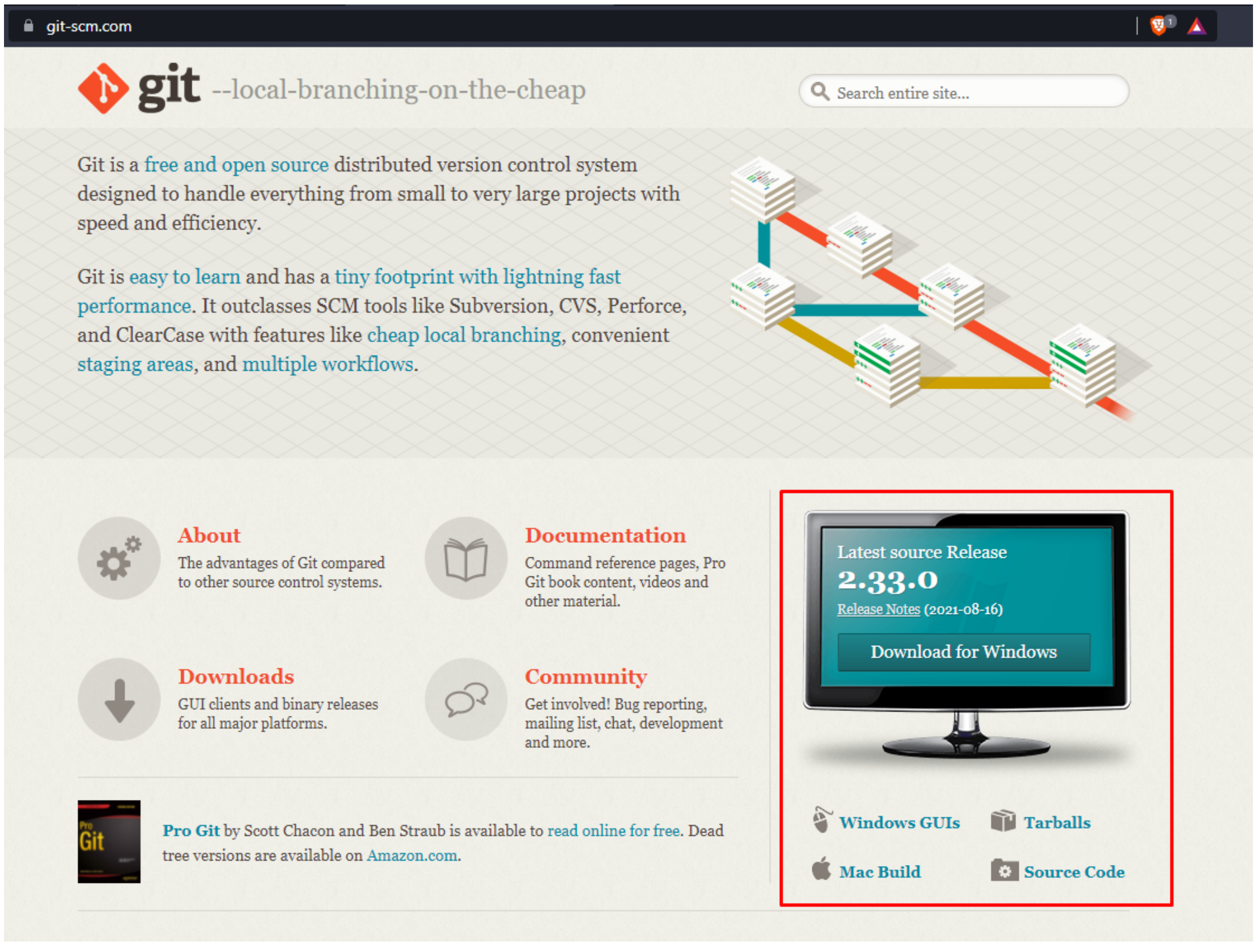
Step 1: Open up your browser and go to https://www.git-scm.com and choose the download option for your operating system. It's a command-line operation for Linux and macOS users, but an installation wizard is available for Windows users.

For Linux:

```
$ sudo apt install git-all
```

For Mac: (You need Homebrew to execute this command. Git also ships with XCode, so you might already have it if you are using XCode)

```
$ brew install git
```



git-scm.com

git --local-branching-on-the-cheap

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About The advantages of Git compared to other source control systems.

Documentation Command reference pages, Pro Git book content, videos and other material.

Downloads GUI clients and binary releases for all major platforms.

Community Get involved! Bug reporting, mailing list, chat, development and more.

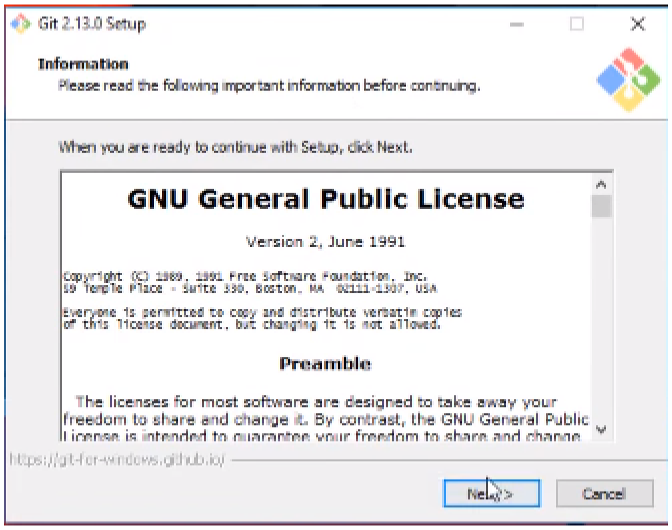
Latest source Release 2.33.0 Release Notes (2021-08-16)

Download for Windows

Windows GUIs Tarballs Mac Build Source Code

Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Step 2 (For Windows users from here on): Open the installer (the .exe file which you downloaded)


Git 2.13.0 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

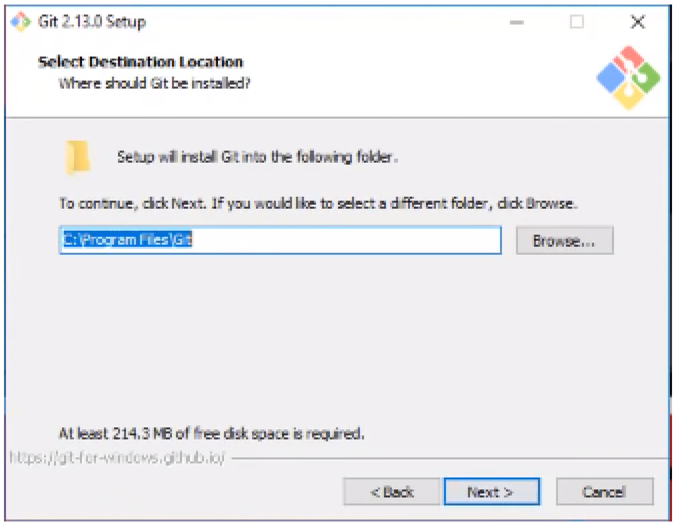
Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://git-for-windows.github.io/

Step 3: Choose the directory in which you want to install Git.

Preferably use the root directory under the C drive, or any directory which does not require elevated permissions, since you might need them when you use git commands from the terminal.


Git 2.13.0 Setup

Select Destination Location

Where should Git be installed?

Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

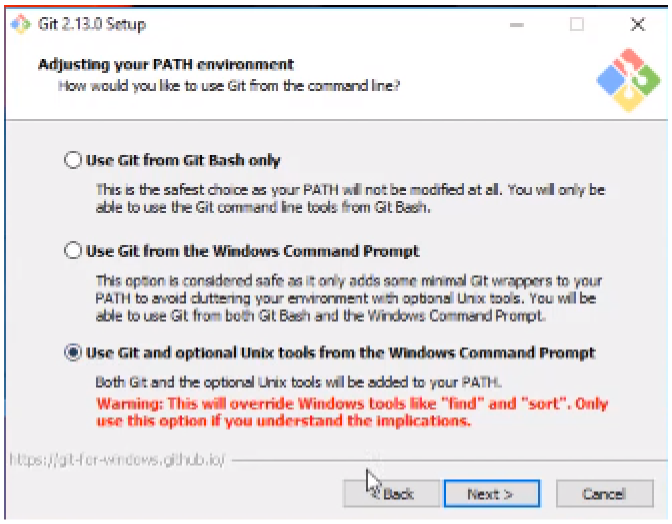
C:\Program Files\Git

At least 214.3 MB of free disk space is required.

https://git-for-windows.github.io/

Step 4: Proceed with default options, create a start folder if you want, until it asks you about *Adjusting the PATH environment*

Choose the second or the third option to be able to use the git CLI from other terminal apps like CMD or PowerShell and choose the third option for Unix commands like ls and pwd.



Git 2.13.0 Setup

Adjusting your PATH environment

How would you like to use Git from the command line?

○ **Use Git from Git Bash only**

This is the safest choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

○ **Use Git from the Windows Command Prompt**

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

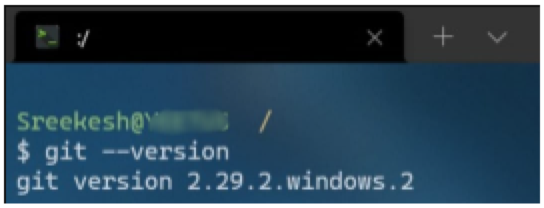
[✓ **Use Git and optional Unix tools from the Windows Command Prompt**]

Both Git and the optional Unix tools will be added to your PATH.

Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

Step 5: Proceed with the installation. Once completed, close the installer. Now, open your terminal (cmd or PowerShell) and enter this command:

```
$ git --version
```



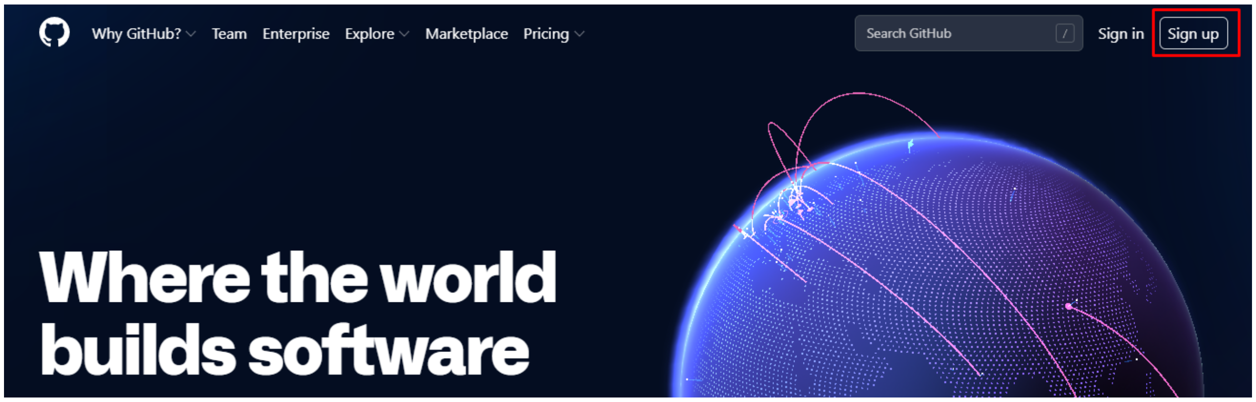
```
Sreekesh@[redacted] /
$ git --version
git version 2.29.2.windows.2
```

If you get back your Git version as the output, Git is installed successfully. You can use git-bash from /bin inside the Git directory.

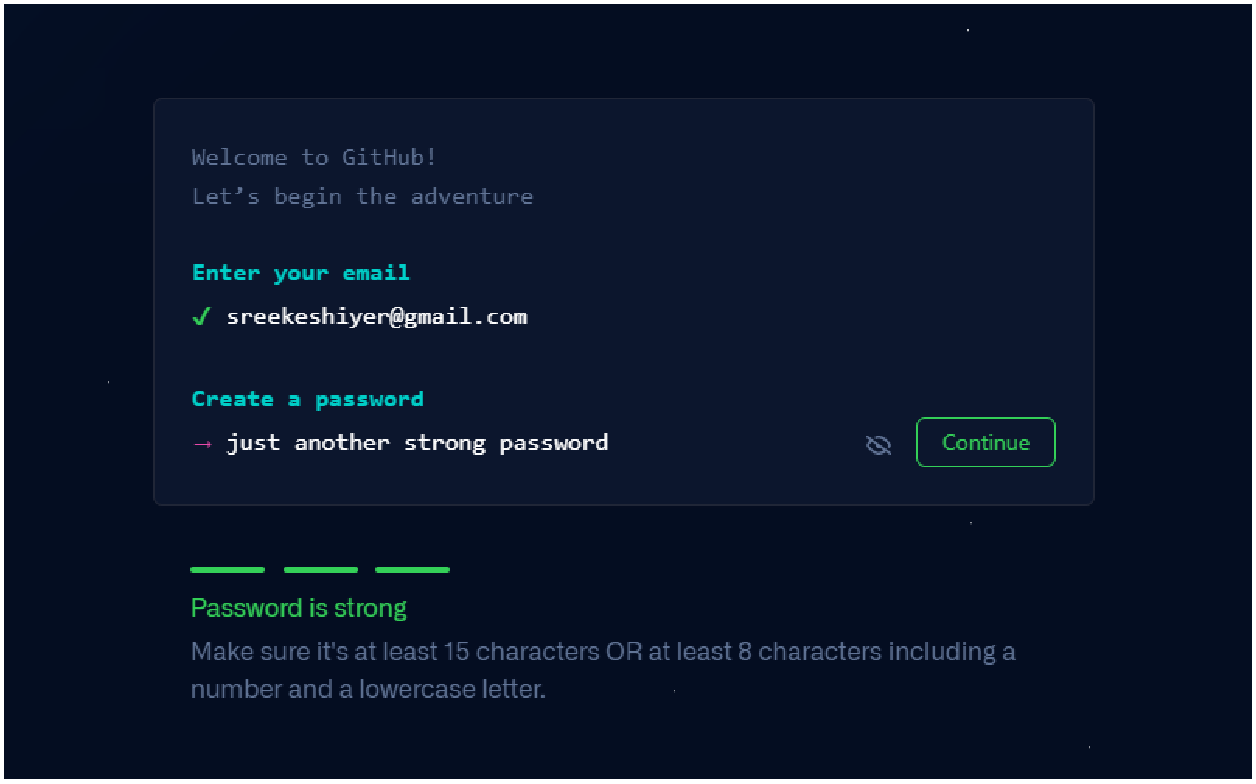
Github:

(Explain steps for creating Github account with screenshots)

Step 1: Go to https://www.github.com/ and choose *Sign Up* on the top right



Step 2: Enter your email address (enter your school/organization-provided email address, you might get a GitHub PRO subscription via GitHub Education) and create a password.



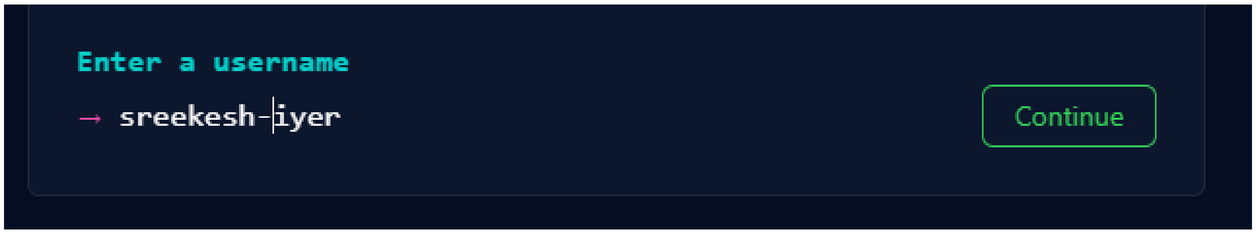
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ sreekeshiyer@gmail.com

Create a password
→ just another strong password

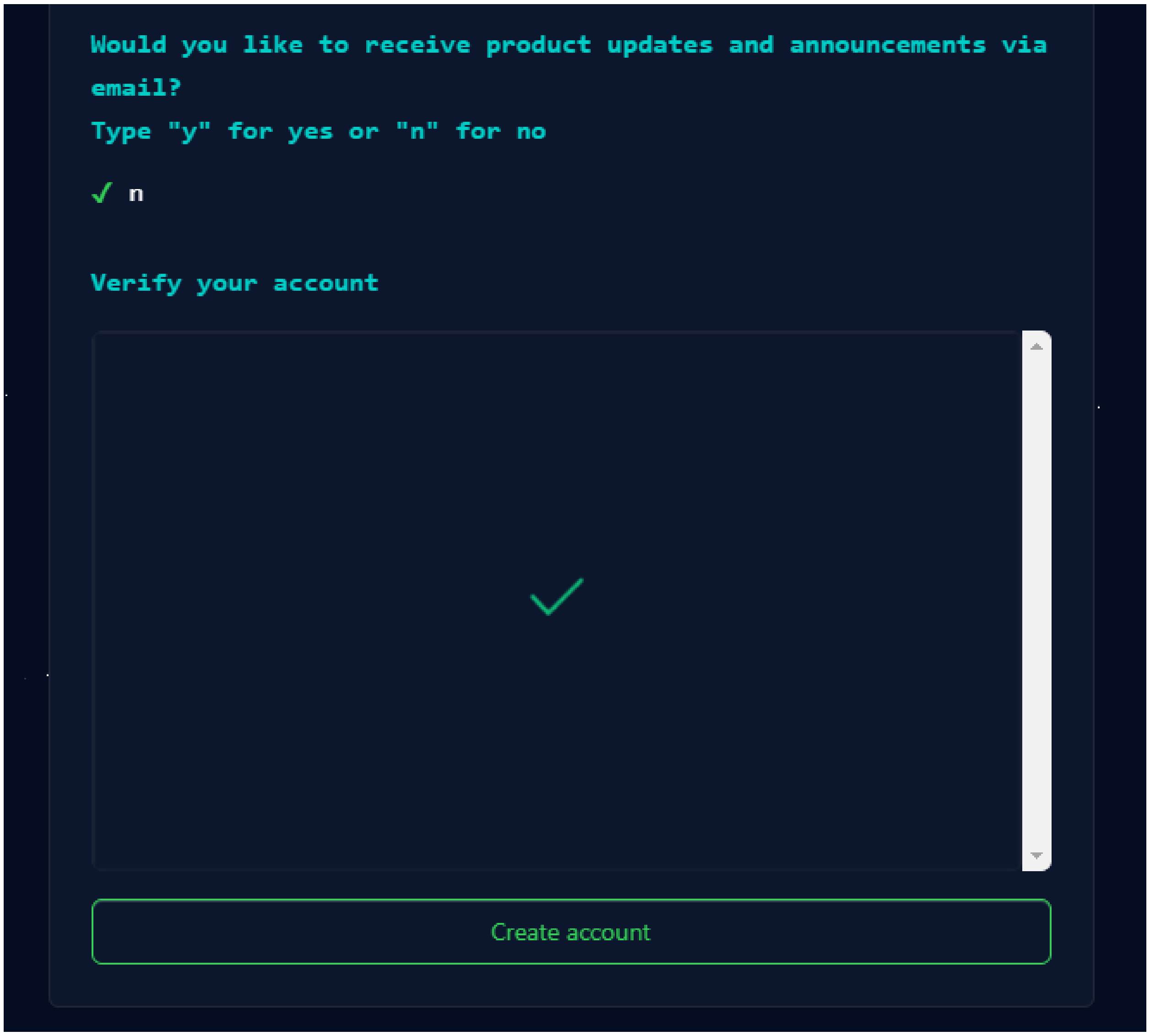
Password is strong
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Step 3: Enter a username for yourself. This is also used as a link to your profile, so people can find you at github.com/<your_username> to check out your repositories. You can proceed if the username you entered is valid and available.



```
Enter a username
→ sreekesh-iyer
```

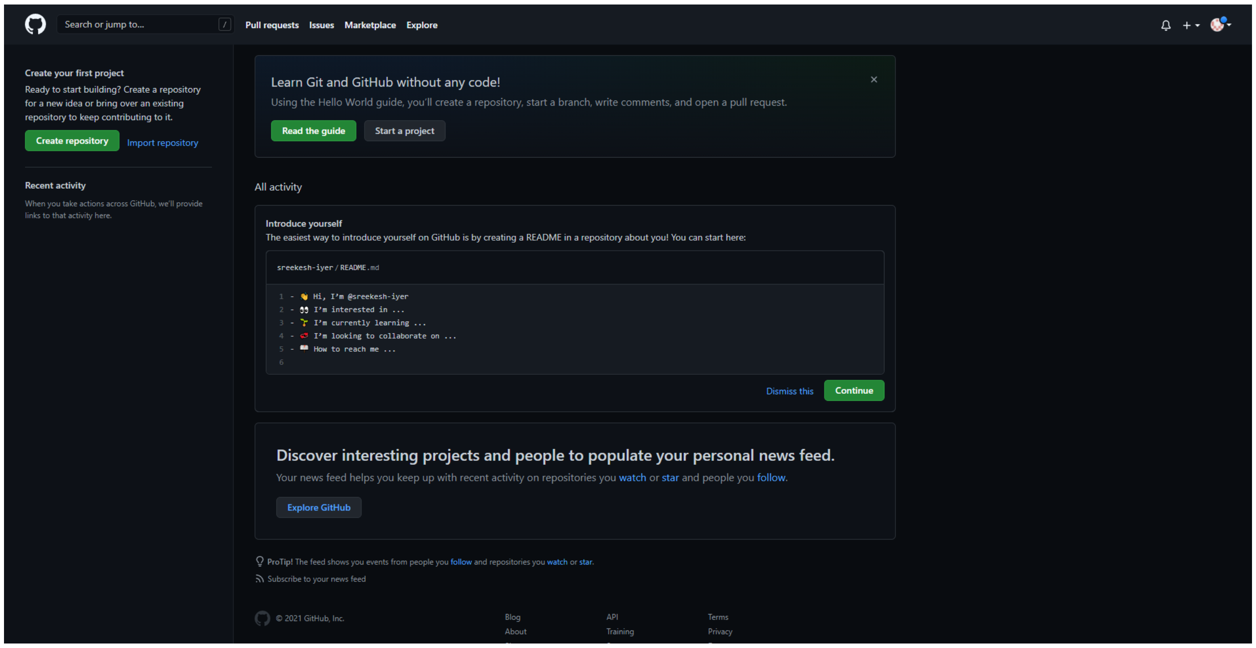
Step 4: You can choose if you want emails from the GitHub newsletter team or not. Then, verify yourself by choosing the correct patterns and click on *Create Account.*



```
Would you like to receive product updates and announcements via
email?
Type "y" for yes or "n" for no
✓ n

Verify your account
```

Step 5: Lastly, verify your email address by entering the code you get from GitHub and optionally, fill up the details for personalization. You can skip this step. Once done, you are redirected to your dashboard.



Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

All activity

Introduce yourself

The easiest way to introduce yourself on GitHub is by creating a README in a repository about you! You can start here:

```
sreekesh-iyer / README.md

- 👋 Hi, I'm @sreekesh-iyer
- 👀 I'm interested in ...
- 🌱 I'm currently learning ...
- 💞️ I'm looking to collaborate on ...
- 📫 How to reach me ...
```

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you watch or star and people you follow.

ProTip! The feed shows you events from people you follow and repositories you watch or star.

Subscribe to your news feed

© 2021 GitHub, Inc.

Conclusion:

In this experiment, we learned about the importance of Version Control, how to install Git CLI and how to create a GitHub account.