A Practical Approach to GIT Using Bitbucket, GitHub and SourceTree

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Subject Area: Computer Science.
Type of the Paper: Experimental-based Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: https://doi.org/10.5281/zenodo.7262771
Google Scholar Citation: IJAEML

How to Cite this Paper:
Chakraborty, S., & Aithal, P. S., (2022). A Practical Approach To GIT Using Bitbucket, GitHub and SourceTree. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *6*(2), 254-263. DOI: https://doi.org/10.5281/zenodo.7262771

International Journal of Applied Engineering and Management Letters (IJAEML)
A Refereed International Journal of Srinivas University, India.

Crossref DOI: https://doi.org/10.47992/IJAEML.2581.7000.0156

Received on: 12/09/2022
Published on: 29/10/2022

A Practical Approach to GIT Using Bitbucket, GitHub and SourceTree

Sudip Chakraborty [1] & P. S. Aithal [2]

[1] D.Sc. Researcher, Institute of Computer Science and Information sciences, Srinivas University, Mangalore-575 001, India,
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2] Vice Chancellor, Srinivas University, Mangalore, India,
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *we use GIT for version control management. It is widely used in teamwork. Standalone developers mostly do not prefer to use it. But in all types of development, we can use it to manage our software efficiently. We demonstrate how we can use the GIT version control in our software project through the step-by-step procedure. We will use Bitbucket and GitHub as GIT servers and SourceTree as GIT clients for the demonstration. The SourceTree has GUI, which we generally prefer. This paper might be helpful who are trying to integrate the GIT system into their project.*

Design/Methodology/Approach: *We create a new remote GIT repo (Repository) in Bitbucket and GitHub. Then install a GIT client into our system. The remote repo is cloned into our system through it. Inside the local repo, we add and modify files. After a good change, we must send it to the Stage. We are writing comments on it and committing to the changes. So, GIT can keep a change record. We can revert to the previous commit point if the current changes are unsuitable.*

Findings/Result: *Here, we demonstrate how to integrate Git easily. Individual researchers can incorporate GIT into their projects efficiently.*

Originality/Value: *Many documents are available on the net for version control system demonstrations. Here we demonstrate in different ways so that anyone can easily understand and integrate these efficient tools.*

Paper Type: *Experimental-based Research.*

Keywords: Global Information Tracker, GIT, Practical example on Git

1. INTRODUCTION :

The software product, especially code writing, is easily manageable using GIT technology. Some researchers do not use it, either not understanding better or not willing to provide extra care to the code management. End of the day, they created a backup copy and kept it inside the system. It has a couple of issues. First, keeping the previous version of the code takes comparatively large space. Second, If the system crashes anyhow, the complete project is lost forever. Third, if the running program is more unstable than the previous one, it needs careful consideration to fetch the previous one. So, the better Option is to use GIT. Initially, it takes some learning curve. But once it becomes familiar, it will be an excellent tool for managing the code. Here we will see the common term available on GIT and its meaning.

GIT Repository- The GIT repository is the working place where the files are stored. It has a file structure to track file changes. In the GIT folder, we generally keep our content inside a dedicated projects folder. These folders are present locally as well as remote GIT server. The local directory is called the local Repository. And the remote is called the remote Repository.

Clone: The clone means to replicate from one to another. We use this command when we want to copy a remote Git repository to our local machine into a folder. It will copy the GIT housekeeping directory structure as well user project.

Push: The push command updates the changes to the remote Repository. Whatever changes inside the local repo are pushed to the remote GIT server.

Pull: The opposite term of Push is pull. The pull means to bring the changes from the remote Repository to the local Repository.
Fetch: The Fetch tells us any change available in the remote. It does not bring the changes to the local repo.
Merge: This is used to joints more than one development history into a single.
Unmerge: It is the reverse action of the merge.
Stagging: after some changes, if we need to commit, the first step is to send the changes to the staging area. Then it is possible to commit.
Unstaged- it is the reverse action of stagging.
Commit- Update the Repository with the latest changes. It will save the latest changes to the local Repository.
Main- It is a primal activity path.
Branch- It is a separate repository inherited from the main branch. It is used to add some features or experiment purposes. The changes inside the branch don't affect the main branch.
Conflict- generally happens in the team environment, where more than one team member works on a project. Team member changes the file from different branches, and GIT cannot merge. It tries to resolve the issue automatically. If unable to resolve, team members are responsible for solving the problem.
Git Ignore: GIT tracks human-created files only. When we create the project inside some IDE, it makes lots of files to run the IDE, like configuration files, binary files, etc., which we should ignore. GIT provides a file called .gitignore. We need to manually add which type of files to ignore so that it executes efficiently.

2. RELATED WORKS :

Spinellis, D. demonstrate the relevance of Git technology. He also shows how we can make it most helpful in the project [1]. Vuorre et al. provide a tutorial on the Git version control system. This is an efficient document to implement in the project. The collaborating research work is most helpful using these tools. These types of tools can manage workflow management with a little extra effort. The novis knowledge of Git can also be integrated easily into their project [2]. Isomöttönen et al. demonstrate challenges and confusion efficiently in learning version control with Git. Based on their teaching experience and survey data collected from the project-based course, the result analysis of the Git learning procedure [3]. German et al. demonstrate to mine the software project to uncover the development history of the software project. They also highlighted the information on distributed version control systems (D-VCS) [4]. In their paper, casino J. et al. proposed an approach to integrating a compilation mechanism and Git system to manage the software versions [5]. All the above research documents are efficient for experienced researchers. For the initial Stage, we feel the initial implementation document should be as practical-oriented and straightforward.

3. OBJECTIVES :

The GIT is an excellent tool for the pure place software project or software part of any project. But some researcher does not prefer to implement the GIT in their research project. We are providing some helpful information so they can understand the importance of the GIT. Using the practical example, we show how we can quickly implement the GIT into our project.

4. APPROACH AND METHODOLOGY :

Figure 4.1 depicts the block diagram of the overall project. The picture is divided into a couple of blocks—the projects initiated from Remote server accounts creation. We will work with two remote servers: Bitbucket and Github. We can work with an existing account if we already have an account. Here we will see how to create the Account on two web servers from scratch. Inside our Account, we will create a new repository.

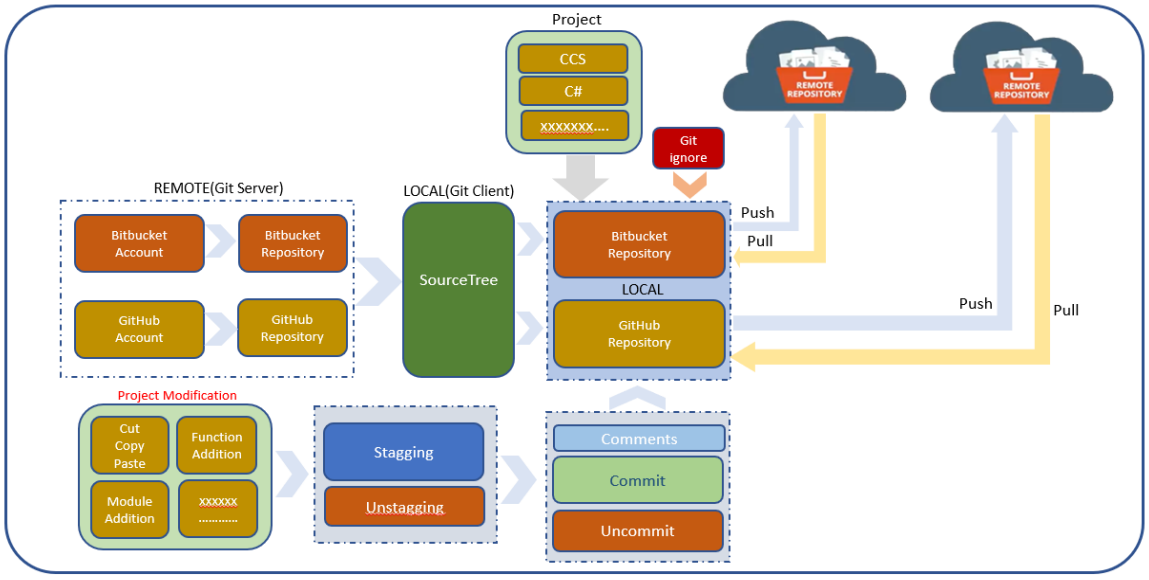

Fig 4.1: Block Diagram of the Project

Git Client: We need a Git client to communicate with the GIT Remote server. Whatever we are doing change, we need to save to the remote server. Due to some incident where the working system is inaccessible, we may lose our work forever, so we need to push a copy to the remote server for safe. To make this task more user-friendly, we need a GIT client. Generally, version control is managed through a command prompt. But for the new user, GUI-based utility makes it more user-friendly. There are lots of software available for GIT clients. We think SourceTree is one of the best user-friendly software to work better. We will install the SourceTree, and clone the remote Repository using it.

Local Repository: After Installing the GIT client, we will create a local repository. It is our local representative which holds our project. We edit our project, which reflects inside the local repo. We usually commit to the local and remote repo at the end of the day or work.

GIT command: The bucket-overflowing GIT commands are available to execute our jobs efficiently. Some commands are enough for the preliminary Stage, like staging, commit, etc.

5. EXPERIMENT :

Now We will experiment to strengthen our knowledge and to be confident in implementing it into our project. First of all, we will practice using Bitbucket and SourceTree.

Bitbucket and SourceTree:

For the experiment, we need to follow the below steps.

1) **Bitbucket account creation**: Open the link in the browser https://bitbucket.org/product. It will Open the Bitbucket website. A **Login** button is on the top right side. Press it. Log in page will open, which depicts in figure 5.1. At the middle bottom of the page, press **Sign up for an account**. Click on it. Press the **Next** button. On the top medium, enter your email address. Type the Email address below, **Sign up for your Account**, and press **Sign up**. The next page shows to check the inbox to log in. if a verification email is unavailable in the given mail-id, pressing **Resend verification email** might be needed. Inside our mailbox, we received from Bitbucket. Press **Verify your email**.

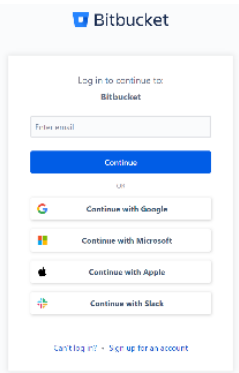

Fig 5.1: Bitbucket login

2) A new page will open. Two textboxes are available for user input. **Enter full name** and **Create a password**. After filling in the textboxes, press the **Sign up** button. The next page will ask us for the User Name in the **Enter username** field. After entering the user name, press the **Continue** button. The user name might not be valid. It will

prompt you to the correct user name if it is already assigned. After a good user name, some questions will be asked. After selecting the response, press the **Submit** button.

3) We are now inside our Account. We can test it out for our login credential details. For that, we need to log out. We need to click the top right side round button, and inside the popup menu, press the **Log out** button below. After logging out again, it will prompt for login. The email is already available inside the text box. Press the **Continue** button. Enter our password and press **Login**. Again we are inside our account page. We confirmed our id and password are correct. We need to keep it in a safe place, which will be required if we forget our id/password or mistakenly log out. Now we successfully created our Account on the Bitbucket website.

4) **New repository Creation:** On the top, we are under the **Your work** tab; in the left middle window, click **Create repository**. It will open a new page. Figure 5.2 depicts the new repository creation page. Enter the project name and repository name. Access level may keep as private. Then press the **Create repository** button. Now Our Repository is created, and the page shows the repository content. It is just a new Repository. Inside the repo, already two files are present. .gitignore and README.md. Copy the path from the address bar and keep it in a notepad. It will require after some time when will clone the Repository. The path looks like this: https://bitbucket.org/drsudipchakraborty/demo/src/master/

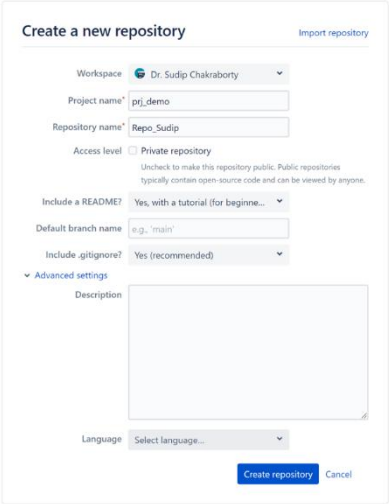

Fig 5.2: New repository Creation Page

5) **SourceTree software Installation**: Download the apps from https://blog.sourcetreeapp.com/2018/04/24/sourcetree-for-windows-enterprise-now-available/. After downloading the setup file, open it by double-clicking. At first registration page will appear, depicted in figure 5.3. Press the **Skip** button. The "**Pick tools to download and install**" page will appear next. Press the "**Next**" Button. The next page is "**Preferences**." On this page, two text boxes appeared. Enter "**Author Name**" and subsequent "**Author Email Address**." Press **next**. The next page will appear, "**Load SSH key?**". Press the "**No**" button. Then SourceTree app will open. On the top left, press the "**Create**" Button. Minimize the app and create a folder anywhere in the system. the Desktop might be OK. Now maximize the app and use the browse button to select the path we created just now. Press the "**Create**" Button. It made local repo.

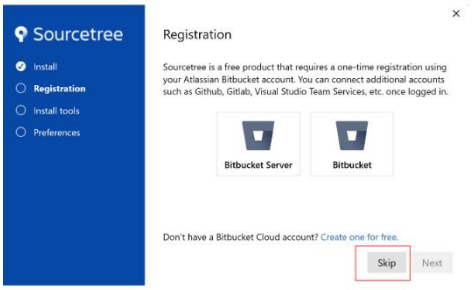

Fig 5.3: SourceTree Installation Page

6) **SSH Key Entry**: to communicate between Bitbucket and SourceTree through secure socket protocol, we need to configure both the Bitbucket server and source tree application. We need one pair of public and private keys. Lots of ways to generate private and public keys. Here we will see how we can create these keys using puttygen. This is free software. The link https://www.puttygen.com/ will open the website to download the application when we are experiencing Putty 0.77 being released. At the top, there is a "**Download Now**" button. Press to download the apps. The download page will appear. Download the installer

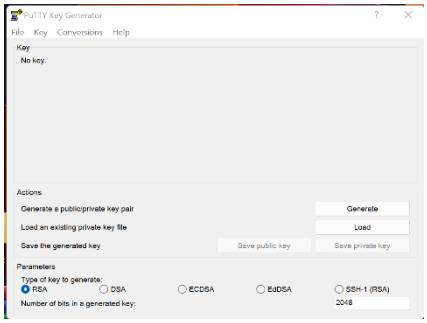

Fig 5.4: Puttygen interface

according to the operating system. We select 64-bit windows installer putty-64bit-0.77-installer.msi. After a successful download, we need to install the application. It is pretty simple. Click on the installer. Press "**Next**." "**Next**"> "**Install**" >" May be asked operations system level authentication> level "**Finish**."

7) From the windows program search button, search "PuTTYgen." Press Open. The software depicts in figure 5.4. now follow the below procedure described in figure 5.5.

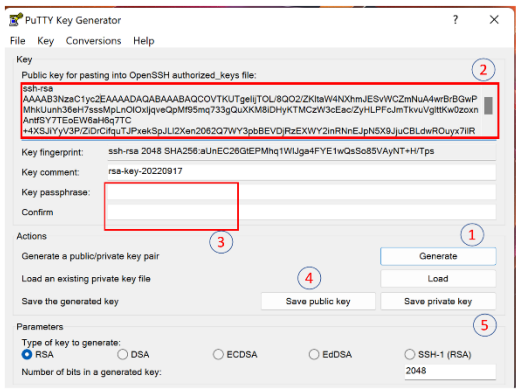

Fig 5.5: Puttygen Interface for Key Generator

   a. **Generate**: right side, there is a "**Generate**" Button. Press it. Keep moving the mouse pointer while the progress bar is ongoing; if the mouse pointer not moves, the progress bar stops. So keep moving. After a while, the generation process will be completed.
   b. **Save Authorized key**: In the figure, mark(2) part, right-click, select all and copy the text. Paste it inside a notepad and save it as authorized_key.txt. It needs to be later.
   c. **Enter passphrase:** In the "**Key passphrase**" textbox(in figure 3), enter a **secret name**, and in confirm textbox, text the same word. We need to keep writing anywhere. It will require after a while.
   d. **Save public key**: Now, "**Save public key**" button and save the file.
   e. **Private key**: Press the "**Save private key**" button and save the private key.
   f. **Apps Close**: Close the PuTTY Key Generator application.

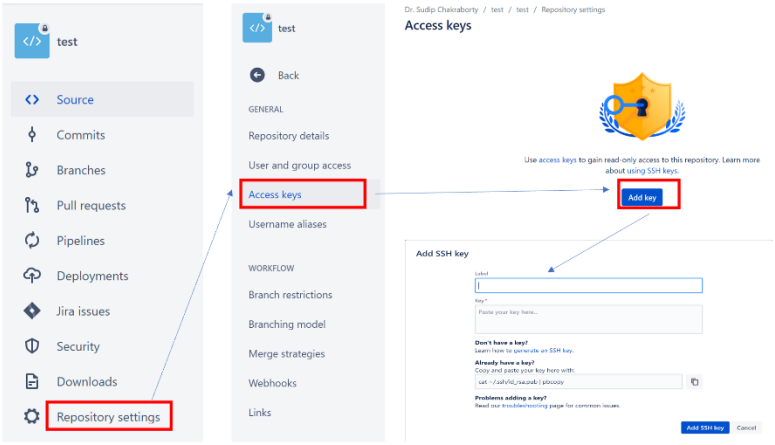

Fig 5.6: Add SSH key inside the Bitbucket remote server

8) **ADD SSH to the remote Repository**: We will add the SSH key to the Bitbucket repository. Log in to Bitbucket> Navigate as figure 5.6. Go to Repository settings> Access keys> add keys>in the label field, type a name> next in the required field> from our saved file, open authorized_key.txt> select all> copy> paste key inside the text box. Now at the bottom, press "Add SSH key."

9) **Add the SSH key to the SourceTree**: Open the SourceTree app. From the **Tools** menu, select **Option**. Under the **General** Tab, there is a group box called **SSH Client Configuration**. On the right side of the **SSH key**, click the browse button and select our private key, which was created earlier. Press the **Open.** Enter the **passphrase** if asked. Press **OK**.

10) **Add Bitbucket account to the SourceTree**: Open the SourceTree. Go to the **remote** tab, described in figure 5.7. Click **Add an account**. It will open a page illustrated in figure 5.8. Press the **Refresh OAuth Token** button. It will open the Bitbucket login page inside the browser automatically. Enter the registered email id and password. Press the **Login** button. If the credential is OK, the web browser will show "**Authentication successful."** Close the browser. Our SourceTree windows will show a green check box with an **Authentication OK** message. Press OK. After a while, Our remote repo account will be added to the SourceTree. Figure 5.9 depicts our Account and lists the available Repository under the Account. Minimize the SourceTree application.

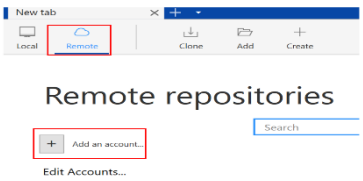

Fig 5.7 SourceTree Remote Tab

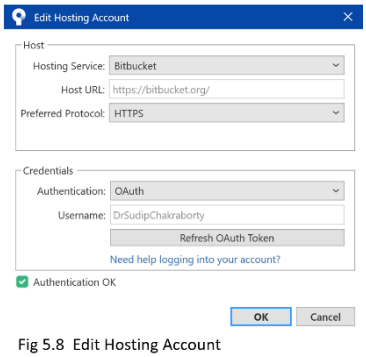

Fig 5.8 Edit Hosting Account

11) **Clone the remote Repository**: Create a folder anywhere in the system. In the application, Navigate to the Clone tab. In the **Source Path/URL** textbox, paste the Bitbucket repo path, which was copied early. Or another way is to log in to Bitbucket> go to **Repositories**> click on our repo name > press the **clone** button. Select **HTTPS**, click the copy button, copy the text box path, and paste it into the application **Source Path/URL** textbox. SourceTree will check the type of Repository. One window will appear—Press the **Select** button. Then the application will show, **This is a Git repository**. Fill the **Destination Path** using the browse button where we created our folder. Press the **Clone** Button. After a while, we will see the remote repo is cloned inside the source tree from Bitbucket remote repository.

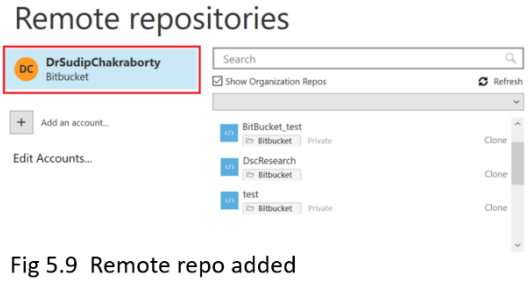

Fig 5.9 Remote repo added

12) **Local repository modification:** Minimize the SourceTree App. Open the folder we created for cloning. Create a text file and add some text. And save it. Close the windows. Maximize the sourcetree application. After a while, we will see some changes, depicted in figure 5.10. Number one, On the top left side, on the "Commit" button's head, "1" is added. It means it detects one change in the local repo. Now needs to commit. Number two, At the bottom of the application, one file is inside the **unstaged files** section. It means it is waiting to send the **stage files** section.

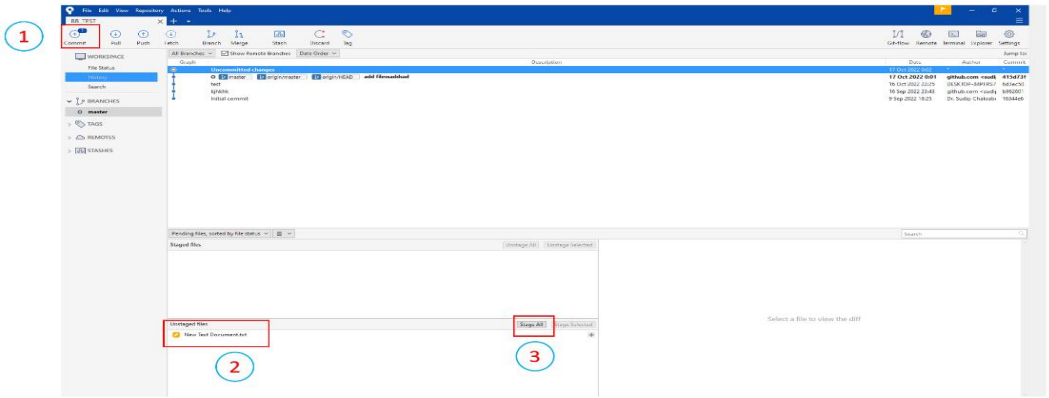

Fig 5.10 SourceTree Commit Step

Number 3, the **Stage All** button. This button is used to send from the unstage to the stage section. Now press the **Stage All** button. The file is now in the **stage** section. Now press the **Commit** button. It will provide a text box at the bottom part of the application. We should write here why we changed the

code or added the file. We may write “project Initialize” or “first commit” like that. After writing the suitable comments, press the **Commit** button at the bottom right side of the application. No mark “1” is on the **Commit** menu button. Instead, mark “1” on the **Push** menu button. It means the local repo is committed and waiting to Commit to the remote repo to keep the change safe. Now press the **Push** button. The progress bar indicates the uploading status. Depending on the internet speed and file, the process generally takes seconds. Now upload is complete. And on the **Push** button, no mark is present. We will follow the steps repeatedly when we realize the changes need to commit. Not always need to Push the remote. But before the system shuts down, we should keep the remote repo updated.
This is the minimal requirement to work on our project using GIT. There are lots of videos available on YouTube on Git. The advanced user can follow. Now we will see how we can work with GitHub and SourceTree.

GitHub and SourceTree

We need to follow the below steps for Github and SourceTree:

1. **Create an Account in Github**: Open https://github.com/. Inside the webpage, On the top right page, Click on the **Sign-up** button. On the next page, as described in figure 5.11, enter email id and press the **Continue** button. Then enter the password. The password needs to maintain some criteria. To strengthen passwords, some requirements, like password length, must be a certain character length, alpha-numeric, at least one capital letter, at least a unique character, etc.

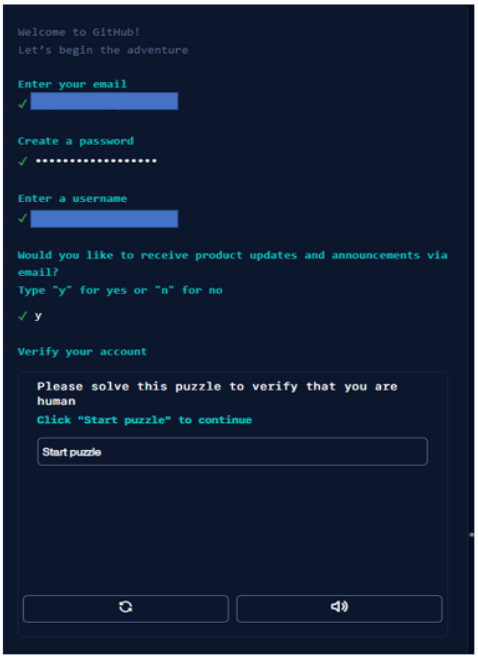

Fig 5.11: Credential entry Page

   If the password is OK, next, enter the user name. Entered user name might be changed if it is already present. Then enter the password. If the password is OK, we will be asked, “would you like to receive product updates and announcements via email? If we want, write ‘y” or ‘n’ for no. " It will verify our Account. Click on “Start puzzle.” It will provide some image identification. Select Some images from mixed images. After this, GitHub will send one code to our email. Paste the verification code. Press the **Create account** button. We will be asked the question, “How many team members will be working with you?” we can say “just me” or “2-5,” or as our choice. Next will be asked, “Are you a student or teacher?” we can click “Student.” Then press the **Continue** button. Then, "What specific features are you interested in using?”. Select our choice and click on “Continue.” Then click “Continue for free.” Now, our Account is created.

2. **Create Repository**: We are now inside the Repository. The top left side is a “+” button—Press **New repository**. The new Repository will open like figure 5.12. Enter the Repository name. Add description. Select “Public” or “Private” Repository. Then press the “Create repository” green-colored button. Our repo is created now.

3. **Add SSH Key inside Github Repository**: click the **profile** button on the top left. One popup menu will appear. Press the **Settings** button. On the next page, on the left side, click on **SSH and GPS keys.** Click on the **New SSH key**. SSH keys/Add new page will appear. In the title box, give a name like "MyKey," etc. paste the authorization key inside the **key** box. Press the **ADD SSH key**. Now SSH key is added.

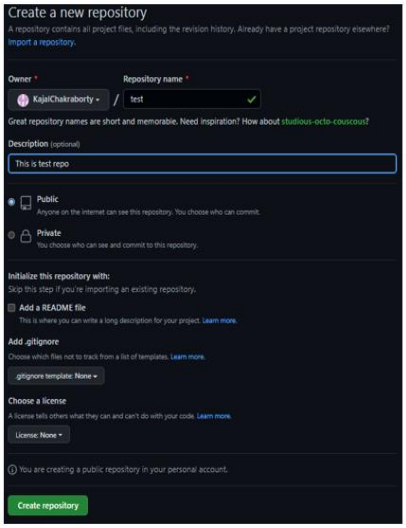

Fig 5.12: New repo creation

4. **Add SSH key to the SourceTree**: Open the SourceTree. Click the **Remote** Tab. Click **Add an account**. Press the **Refresh OAuth Token** button. It will automatically open the GitHub login page. After entering username and password, press the **Sign in** button. Press **Authorize Atlassian** and finally, **Authentication Successful** message will be displayed. Now inside the application created GitHub account will be visible. Now we can clone, create or add a Repository.

Clone Repository: This is used to clone the Repository. We assume that the repo is already present inside the remote Repository, i.e., inside Github. Create a folder where we will clone the remote repo anywhere in the working system. Let's say we create a folder "test" on the Desktop. Maybe a Desktop is good. Now Open the SourceTree. Click on the **Clone** Button described in figure 5.13. Open GitHub using a login credential. Open the repo which we want to clone. Go to the code, then press the button to copy the link. Now go to the SourceTree **Clone** page. Fill the Remote and Local folder paths and press the "**Clone**" Button. Then the cloned project will be shown inside the SourceTree Application after a while.

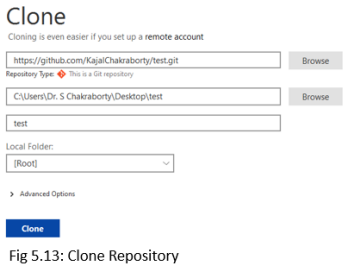

Fig 5.13: Clone Repository

Create Repository: Now, we will see how to create a new repository. Create a folder where we want to keep the Repository. The Desktop may be a good choice. On the Desktop, create a folder. Name it "Demo." Open SourceTree app. On the top menu bar, click **Create** Menu. Click the browse button and select the folder path. If we want to create only local repo, do not check to **Create Repository On Account**. Else, check Option. It will ask for another couple of options. Select **Account** and **Owner**, and give a name for the repo. Then if it is private, check on" Is Private" or keep unchecking. Finally, click on **create**, depicted in figure 5.14. Then it will create a repo for us and display it inside the apps.

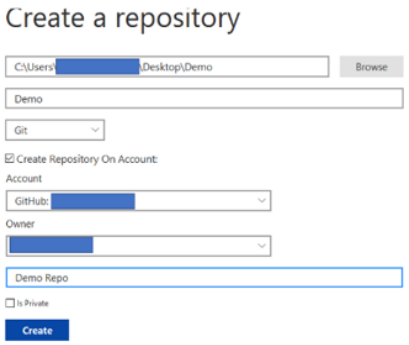

Fig 5.14: Create Repository

Add Repository: we can add an existing Repository inside the SourceTree app. Open the App. From the top Menu bar, click the **Add** Button. One page will open like figure 5.15. The first textbox is for the Working Copy path. We use the browse button to select the active project. Then press the **Add** button. The repo will add to the app.

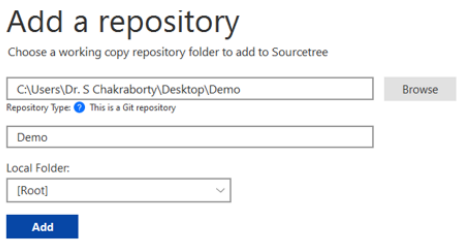

Fig 5.15: Add Repository

After creating or cloning the Repository, we can commit local and remote repo, which is already described before.

6. RECOMMENDATIONS :

- The source tree apps can be downloaded from https://www.sourcetreeapp.com/
- For more details about the git terminology: https://www.geeksforgeeks.org/30-most-important-git-terminologies-that-developers-must-know/

7. CONCLUSION :

We demonstrated how to integrate Git technology into our project. We have gone through step by step using Bitbucket, GitHub, and SourceTree with the practical example. The new researcher who wants to incorporate the GIT technology into their project can find some helpful information.

8. REFERENCES :

[1] Spinellis, D. (2012). "Git," in *IEEE Software*, 29(3), 100-101. DOI: 10.1109/MS.2012.61. Google Scholar↗

[2] Vuorre, M., & Curley, J. P. (2018). Curating research assets: A tutorial on the Git version control system. *Advances in Methods and Practices in Psychological Science*, *1*(2), 219-236. Google Scholar↗

[3] Isomöttönen, V., & Cochez, M. (2014, June). Challenges and confusion in learning version control with Git. In International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (pp. 178-193). Springer, Cham. Google Scholar↗

[4] German, D. M., Adams, B., & Hassan, A. E. (2016). Continuously mining distributed version control systems: an empirical study of how Linux uses Git. Empirical Software Engineering, 21(1), 260-299. Google Scholar↗

[5] Casquina, J. C., & Montecchi, L. (2021, September). A proposal for organizing source code variability in the git version control system. In Proceedings of the 25th ACM International Systems and Software Product Line Conference-Volume A (pp. 82-88). Google Scholar↗

[6] Lawrance, J., Jung, S., & Wiseman, C. (2013, March). Git on the cloud in the classroom. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 639-644). Google Scholar↗

[7] Swicegood, T. (2008). Pragmatic version control using Git. Practical Version Control Using Git, 1-184. Google Scholar↗

[8] Schreiber, A., & De Boer, C. (2020, June). Modeling knowledge about software processes using provenance graphs and its application to git-based version control systems. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (pp. 358-359). Google Scholar↗

[9] Shirey, R. G., Hopkinson, K. M., Stewart, K. E., Hodson, D. D., & Borghetti, B. J. (2015, January). Analysis of implementations to secure Git for use as an encrypted distributed version control system. In 2015 48th Hawaii International Conference on System Sciences (pp. 5310-5319). IEEE. Google Scholar↗

[10] Blauw, F. F. (2018, May). The use of Git as version control in the South African software engineering classroom. In 2018 IST-Africa Week Conference (IST-Africa) (pp. Page-1). IEEE. Google Scholar↗

****