

Version Controlling with Git - I

Introduction

By far, the most widely used modern version control system in the world today is Git. 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. A staggering number of software projects rely on Git for version control, including commercial projects as well as open source.

You will learn more on the concepts on version controlling and git in PAF lecture sessions. This lab is a gentle introduction to some basic concepts in Git.

Steps – Part I (Local Repository)

1. Install and configure Git on your computer.
 - For Official Documentation on Git Installation on Windows/ Linux/ Mac follow [this link](#).
2. Once the Installation is complete, open the Git bash if on Windows. On Linux/ Mac, simply open a terminal window. Git bash is a special terminal for Windows only.

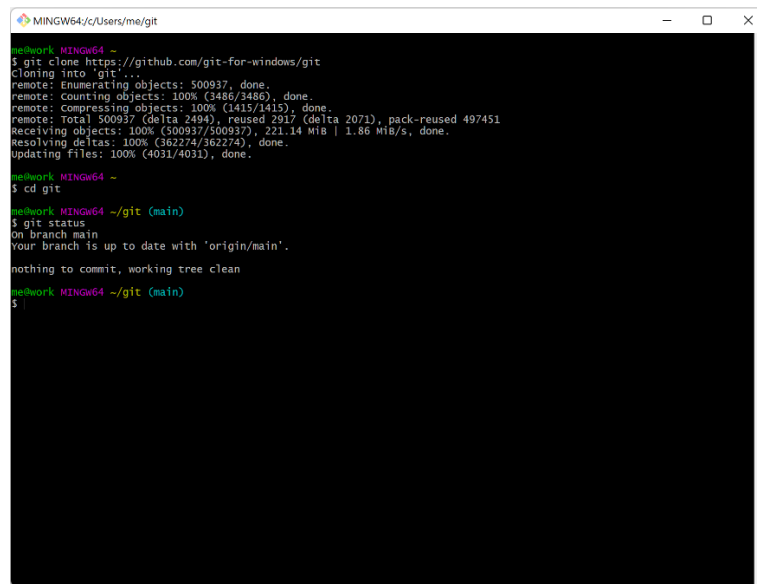


Figure 1: Git Bash on Windows

3. The next step in adding Version Control with git to a project is to initialize the repository.
 - Create a new folder for your repository on your machine.
 - Then inside this new folder, to create a new repo, you'll use the git init command. git init is a one-time command you use during the initial setup of a new repo.

- Executing this command will create a new .git subdirectory in your current working directory. This command is used to start a new repository.

Usage: `git init <project directory>`

4. Next, create HelloWorld.html inside the newly initiated git repository.

```
<html>

  <head>

</head>

  <body>

    <h1>Hello World</h1>

  </body>

</html>
```

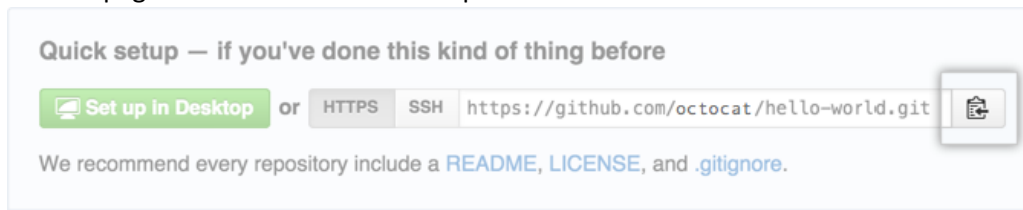
5. Next step is to add the created file to the local repository. For this purpose, git add command will be used. This command adds a change in the working directory to the staging area of the local repository.
 - Usage: `git add <filename>` (to add a specific file)
 - Usage: `git add --all` (to add all added/ modified/ deleted files in the repository)
6. You can use the command `git status` to see the status of the working directory. It is a good habit to periodically check the status of your repository. Execute this command now and observe the output. Try to understand what it means.
7. Finally, once the new/ modified file(s) is/ are added to the staging area, next task to do is to commit them to the local repository. Commit command records or snapshots the file permanently in the version history.
 - Usage: `git commit -m "Type in the commit message"`
 - It is a good practice to write meaningful commit messages so that you and others who refer to your repository later can understand what changes/ additions you made in this particular commit.
 - Once this step is done, your file is now being tracked by Git. Anytime a change happens to this file from now onwards, git will notice and you can effectively keep a check on how the file is modified.

IT3030 – Programming Applications and Frameworks**2025****Steps – Part II (Remote Repository)**

Remote repositories are versions of your project that are hosted on the Internet or network somewhere. To be able to collaborate on any Git project, you need to know how to manage your remote repositories. Let's see in this small activity, how to push your local repository to a remote repository so you can collaborate with others as well.

Github is one well known space to host remote repositories. So, we will Github for this example. But you could try the same with other providers such as BitBucket, GitLab and Azure DevOps and the process should be more or less similar.

1. First login or sign-up to Github.
2. Then create an empty repository on Github following instructions on [this documentation](#). (You may create a Public or Private type repository).
3. Then you need to specify that this remote repository on Github is linked to the local repository on your personal computer. To do so, follow the instructions [here](#).
 - a. The remote url for the GitHub Repository you created can be found on that Repository's page itself. Select the HTTPS option for now.



4. Then follow the instructions available [here](#) from step 9. Steps 1-8 have all been completed in the previous steps followed in the practical sheet.
 - a. You may be asked to enter your Github credentials when you do above.
 - b. However, entering your password wont work here. Instead you will have to use a Personal Access Token (PAT). Follow this [documentation](#) to setup the PAT. Make sure this PAT is copied and saved somewhere safe. It will only be visible once.
 - c. Then try step 9 again, and when prompted for password enter your generated PAT instead.
 - d. If all went well, your code should now be pushed and be available on GitHub.