**Task 1: WORKING WITH GIT**

**GIT:**

Git is a powerful, distributed **Version Control System (VCS)** and **Source Code Management (SCM)** tool designed to track changes in code, facilitate collaboration, and ensure project integrity. Developed by **Linus Torvalds** in 2005, Git is now a cornerstone of modern software development, used by teams worldwide to manage everything from small projects to enterprise-scale applications.

**What is Version Control?**

**Version control is the practice of tracking and managing changes to software code over time.**

Git provides:

1. **Change History:** A detailed log of who made changes, what was changed, and why.
2. **File Versioning:** Enables developers to save, revisit, or restore specific versions of code.
3. **Collaboration without Conflict:** Tracks contributions from multiple team members without overwriting or losing work.

**What is Source Code Management (SCM)?**

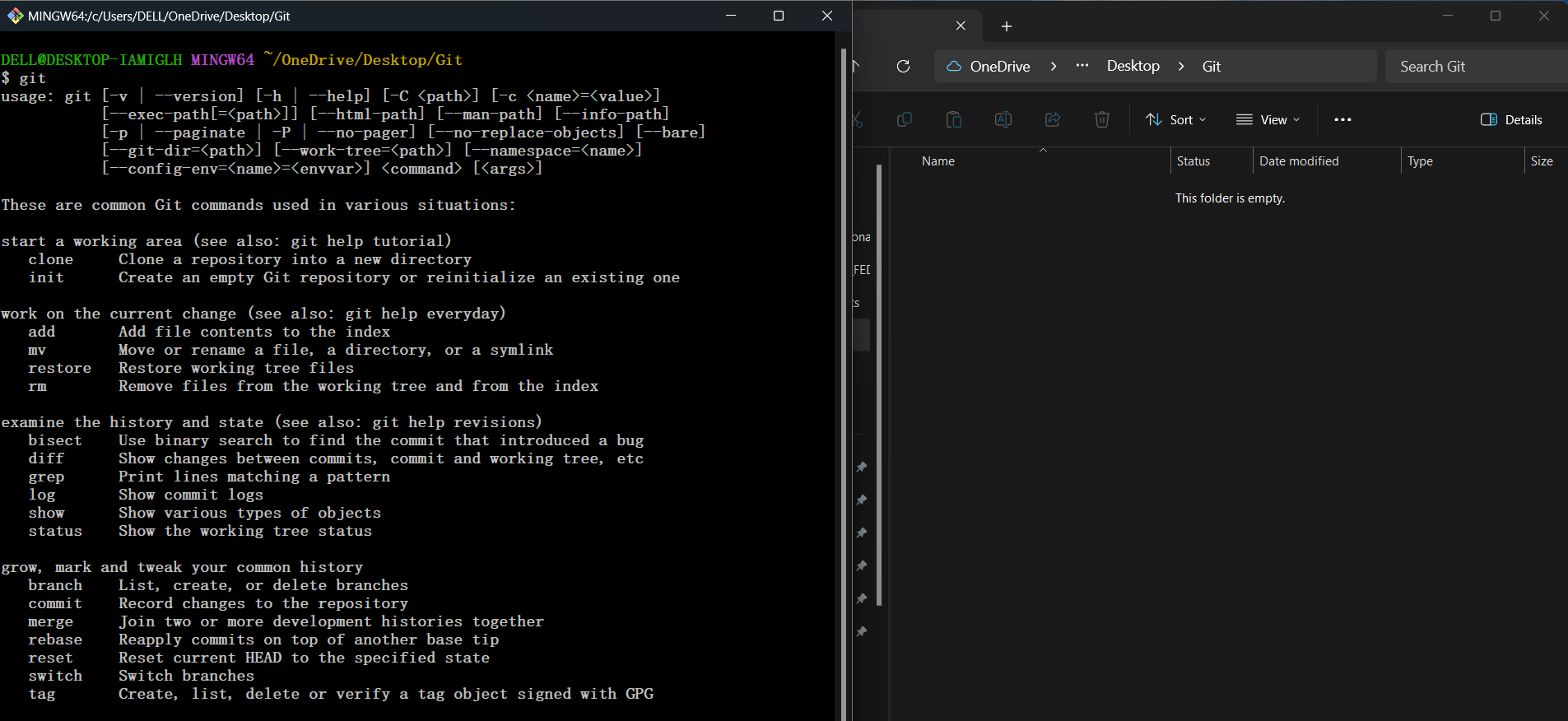
SCM refers to the methodologies and tools used to manage the source code of a project. Git’s SCM capabilities allow developers to:

* Maintain a single source of truth for the entire team.
* Manage multiple development streams through branching and merging.
* Automate workflows with integration into Continuous Integration/Continuous Deployment (CI/CD) pipelines.

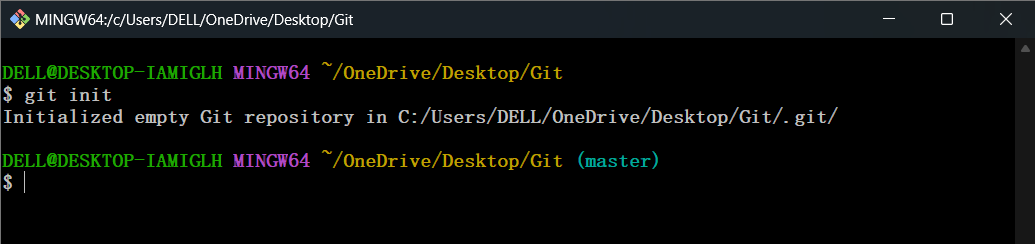
**Key Features of Git**

* **Distributed System:** Every user has a complete copy of the repository, enabling offline access and faster operations.
* **Version Tracking:** Keeps a detailed history of changes, allowing you to review or revert to previous versions.
* **Branching and Merging:** Encourages experimentation with lightweight branching and easy merging of code.
* **Collaboration-Friendly:** Multiple developers can work on a project simultaneously without overwriting each other’s changes.
* **Efficiency:** Optimized for speed and efficiency, even with large projects.

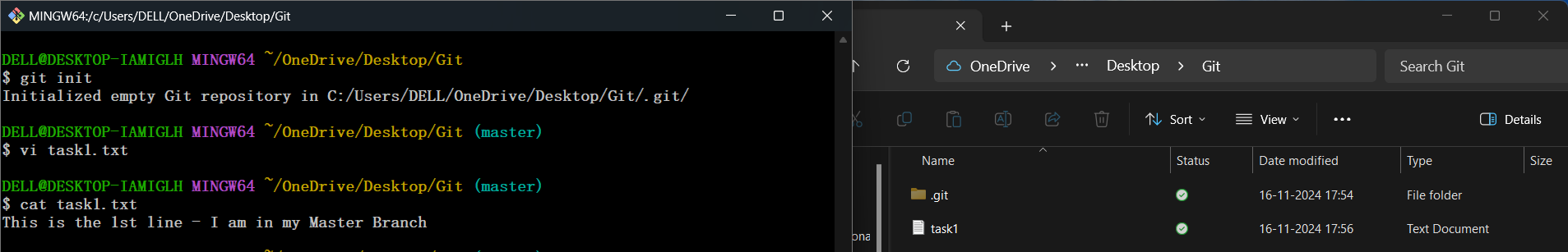
1. **Installed GIT on my local machine**



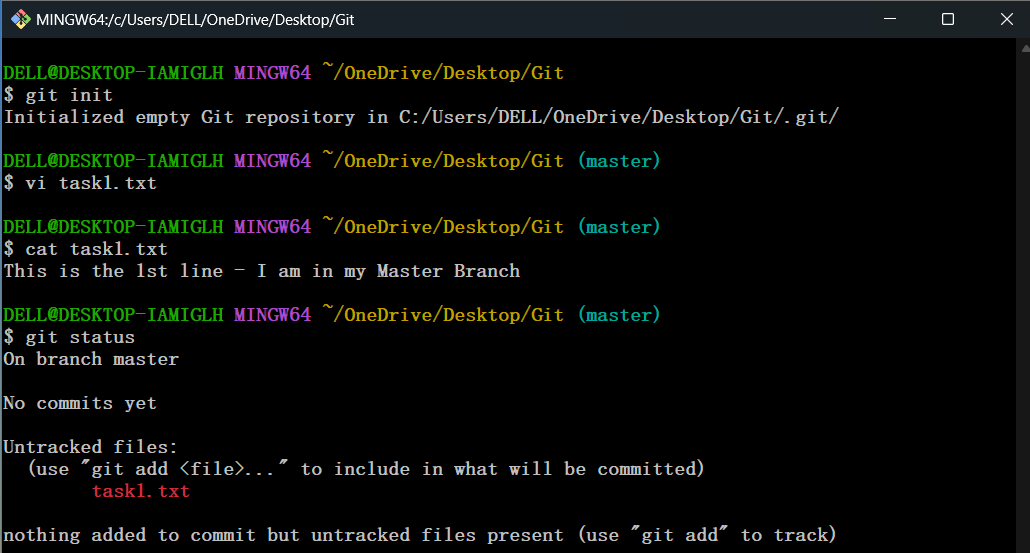
1. **New Repository Created using the command “ $ git init “**



1. **Created a file using “$ vi task1.txt” command on the Master Branch**

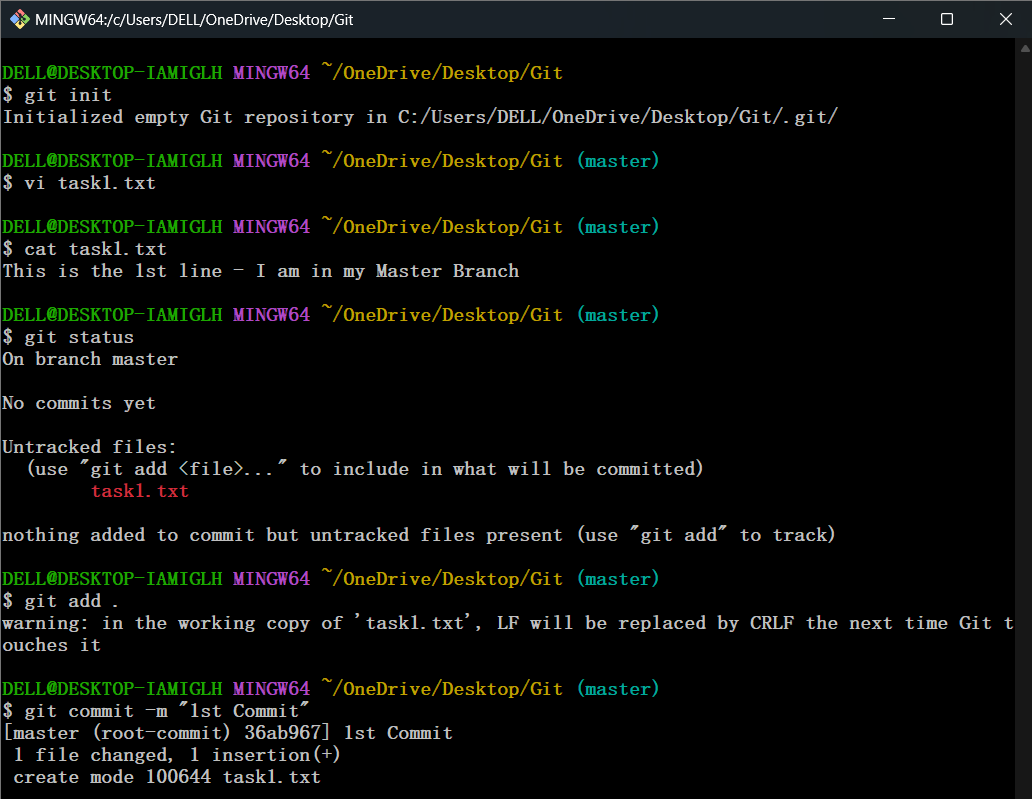


1. **Checked the status using “$ git status” command**

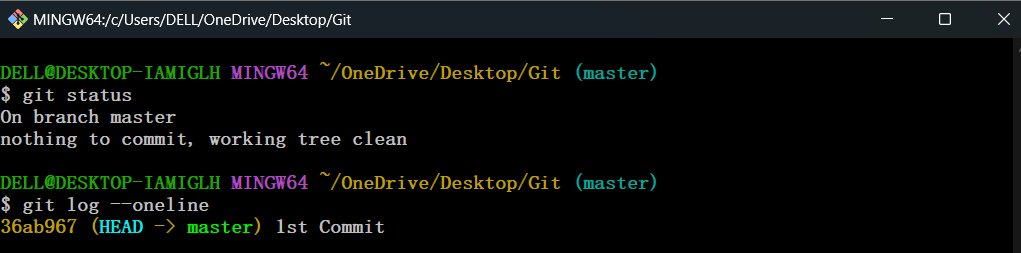


1. **Done Add and Commit using**

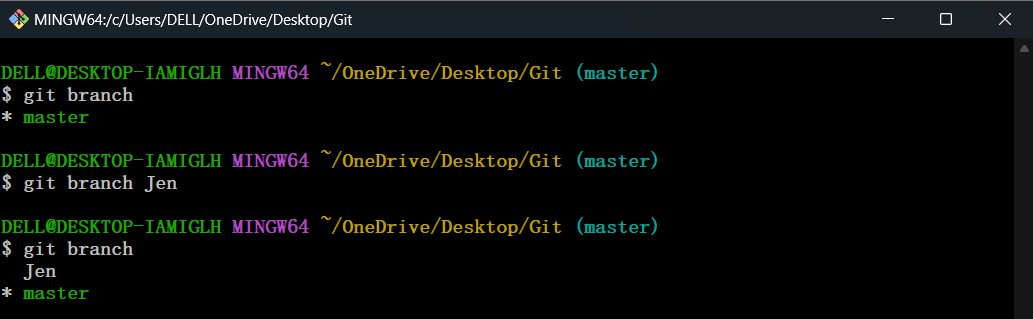
**“$ git add .” & “$ git commit –m” commands**



1. **checked the log using “$ git log --oneline” command**

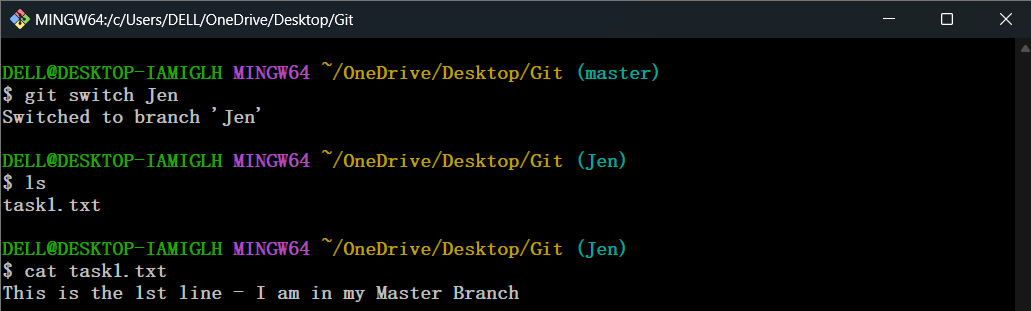


1. **Listed out the Branches using “$ git branch“ command & Created a New Branch(Jen) using “$ git branch Jen” command**

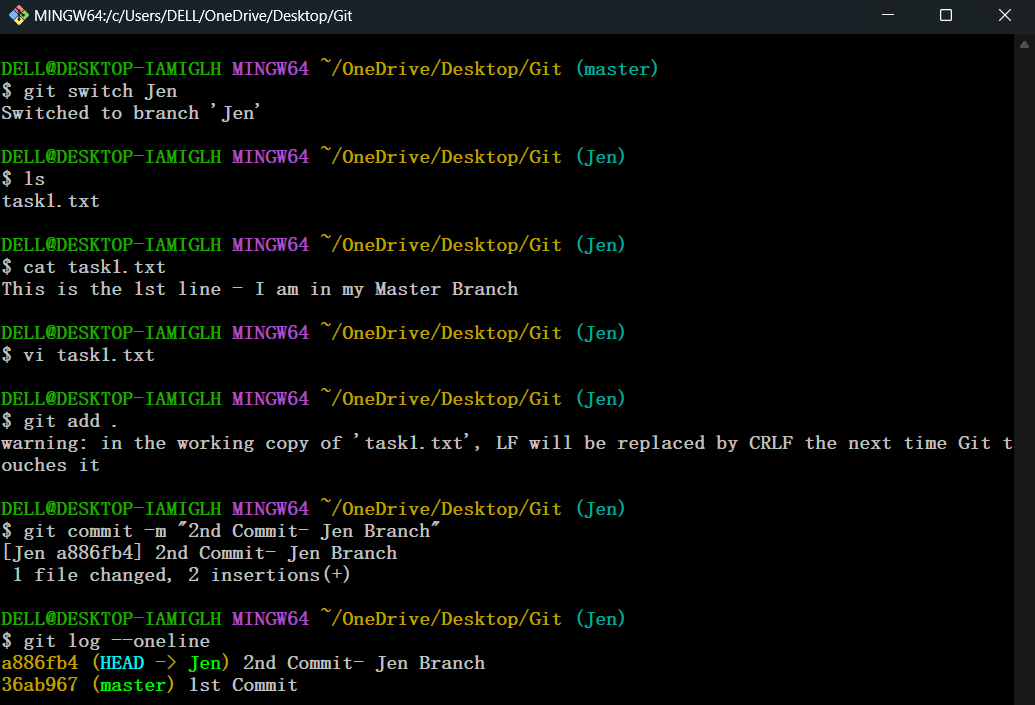


1. **Switched into the newly created branch using**

**“$ git branch Jen” & Checked the reflected file which is created in the Master Branch**



1. **The 2nd commit done in the Jen Branch & Checked the log**



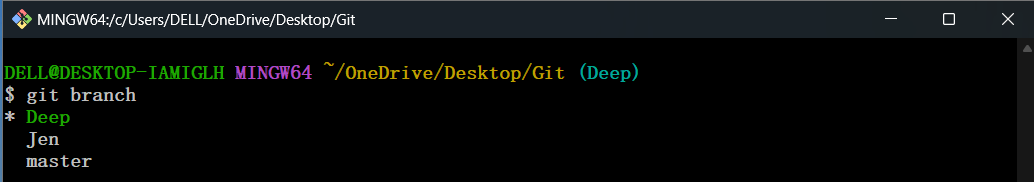
**10. Switched into Master Branch & Checked the Content in the file1.tx**



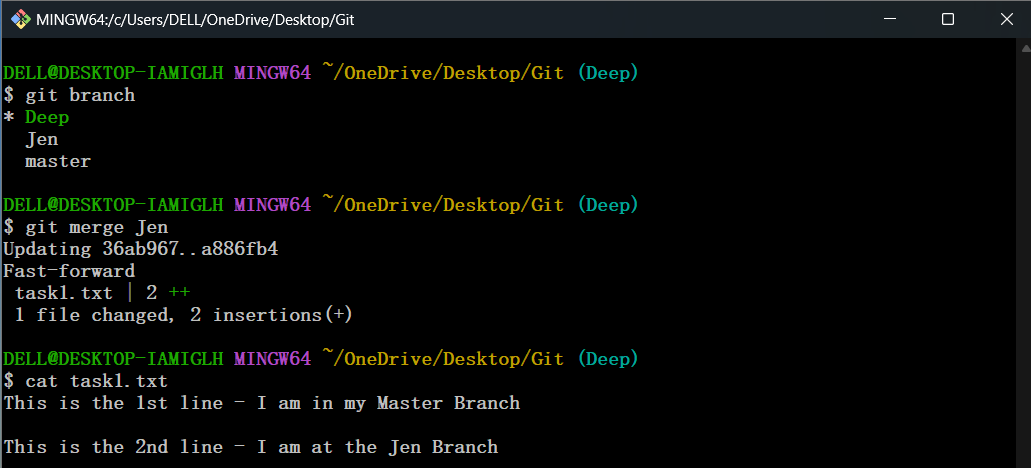
-> **Steps 9 and 10** illustrate that the files in the master branch are included in feature branches when they are created. However, changes made in feature branches will not be reflected in the master branch unless they are merged. This behavior is called **branch divergence**.

-> It describes how changes in one branch (such as the master branch) do not automatically propagate to other branches (such as feature branches) unless explicitly merged or rebased. **This allows each branch to develop independently, maintaining a clean and isolated history of changes.**

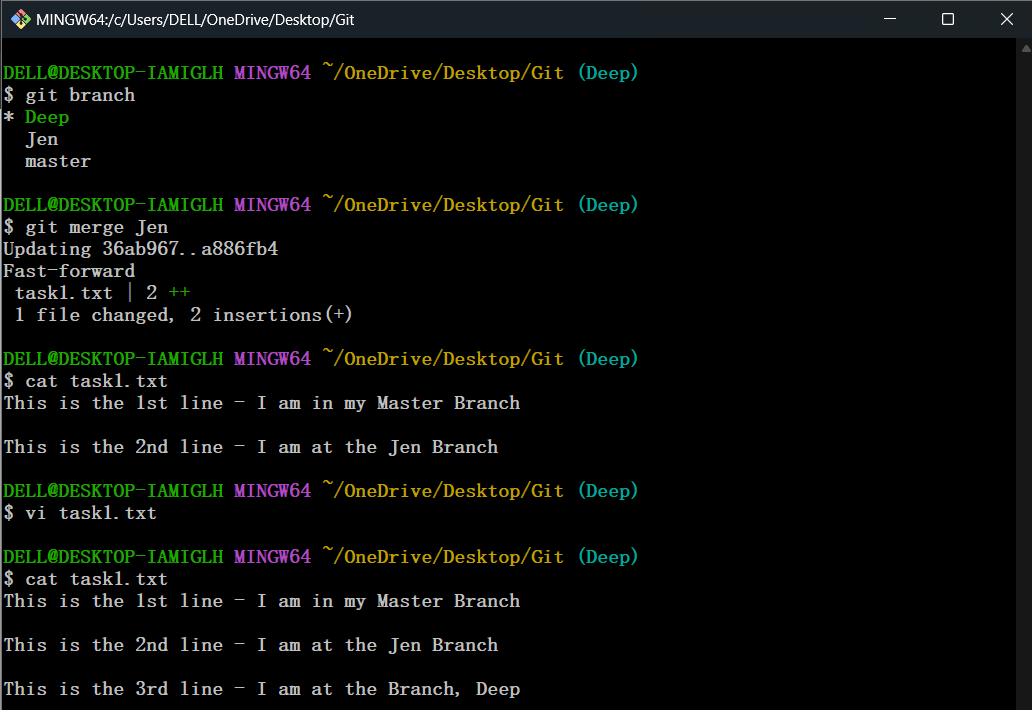
**11.A new branch Deep created & Listed out the Branches created as of now**



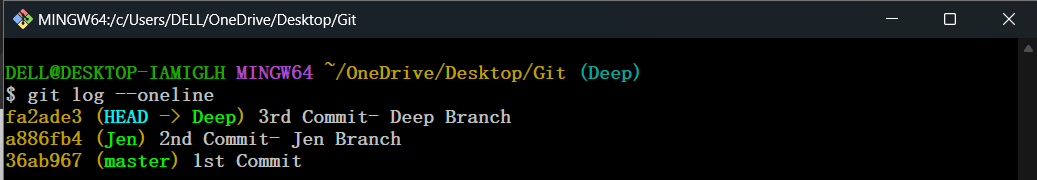
**12. Merged Jen branch using “$ git merge Jen” on the base branch Deep**



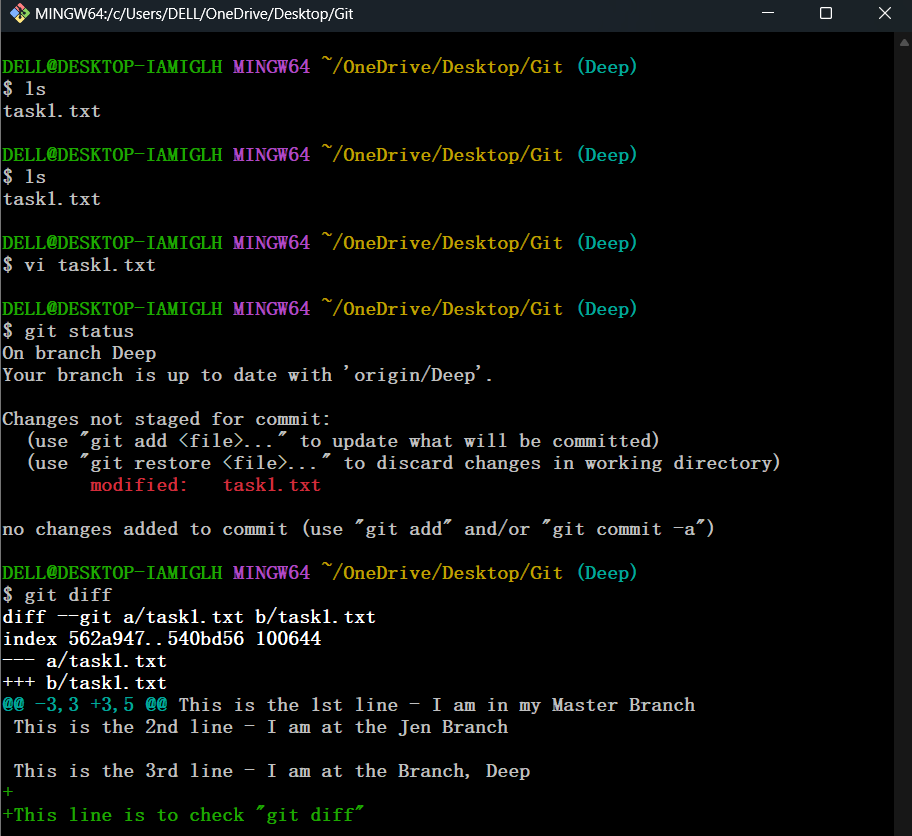
**13. 3rd line added in the file1.txt file in the Deep Branch**



**14. Checked the log in Deep Branch**

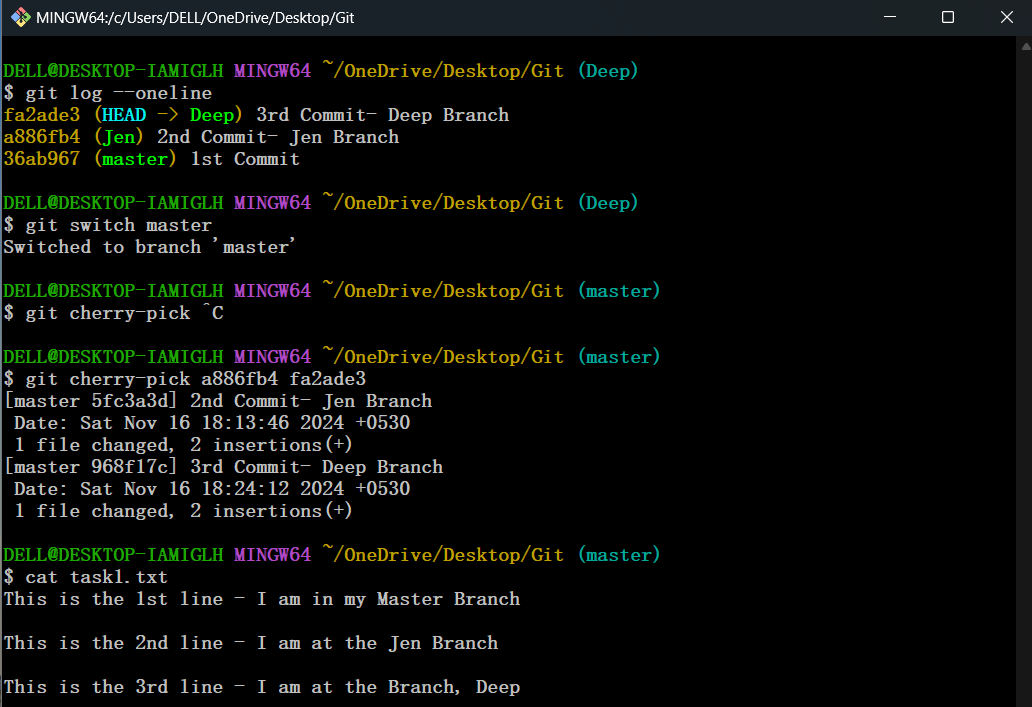


**15. Detected the differences that are not committed yet, using the command “$ git diff”**



The **git cherry-pick** command is used to apply specific commits from one branch to another without merging the entire branch. It allows you to selectively bring changes from a commit into your current branch.

**16. Using “$ git cherry-pick <Commit-Hash>” command the master branch updated with a new commit**



**17. Configurations has done using**

**“$ git config –global user.name “<UserName>” &**

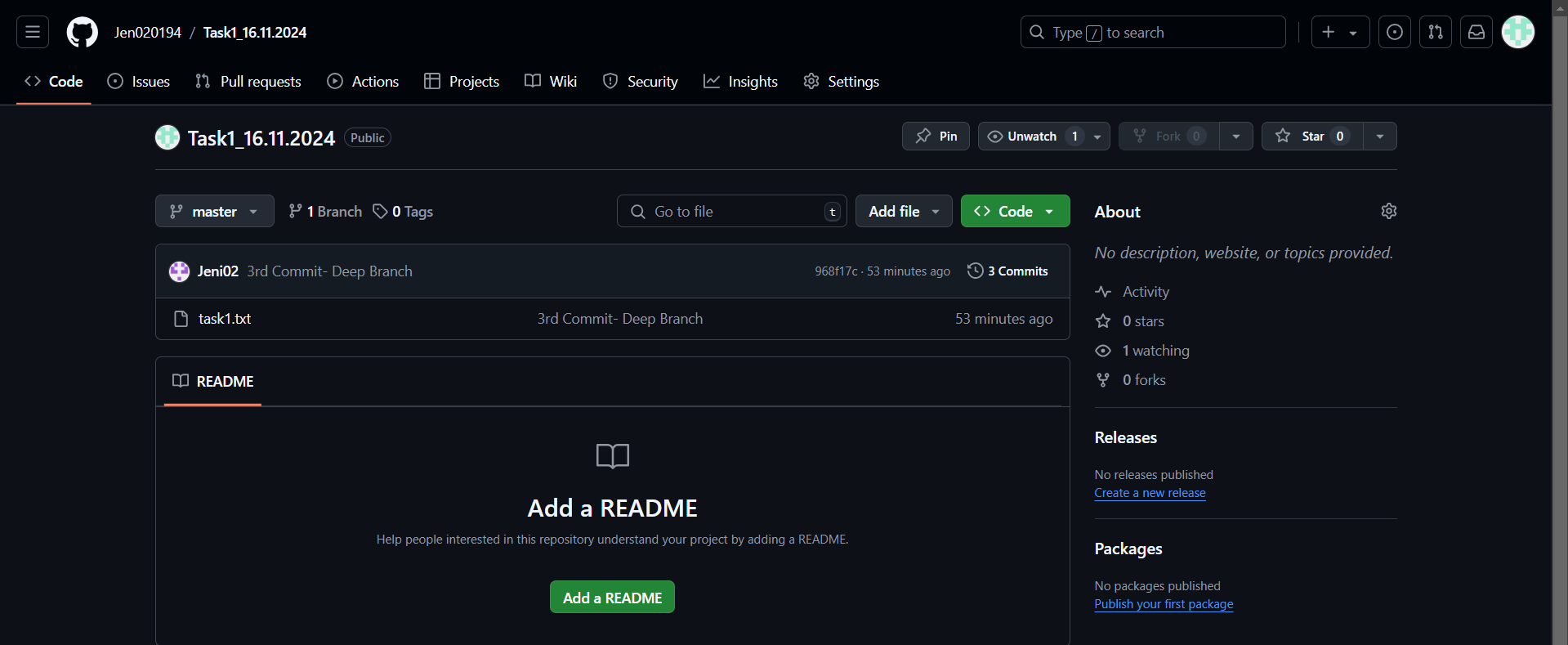
**“$ git config –global user.email “<Mail\_ID>” commands**



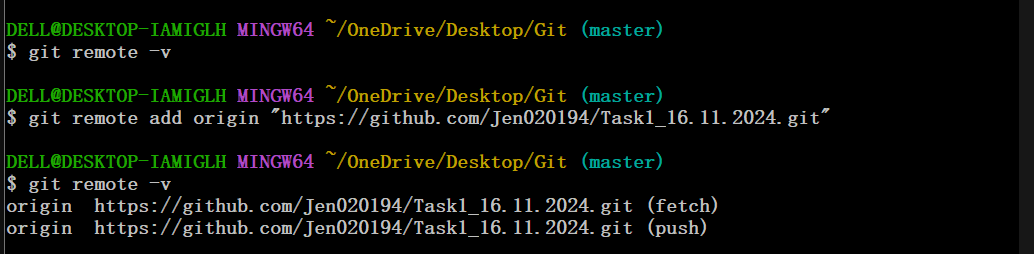
 **Git** is a tool that enables version control by allowing developers to track changes, manage different versions of code, and collaborate on projects. It operates locally on your machine, and doesn't require an internet connection for basic operations.

 **GitHub** is an online service that hosts Git repositories, providing a central location for developers to collaborate on projects. It enhances Git’s functionality with features like issue tracking, team management, and code review through pull requests, making it easier to share and contribute to code globally.

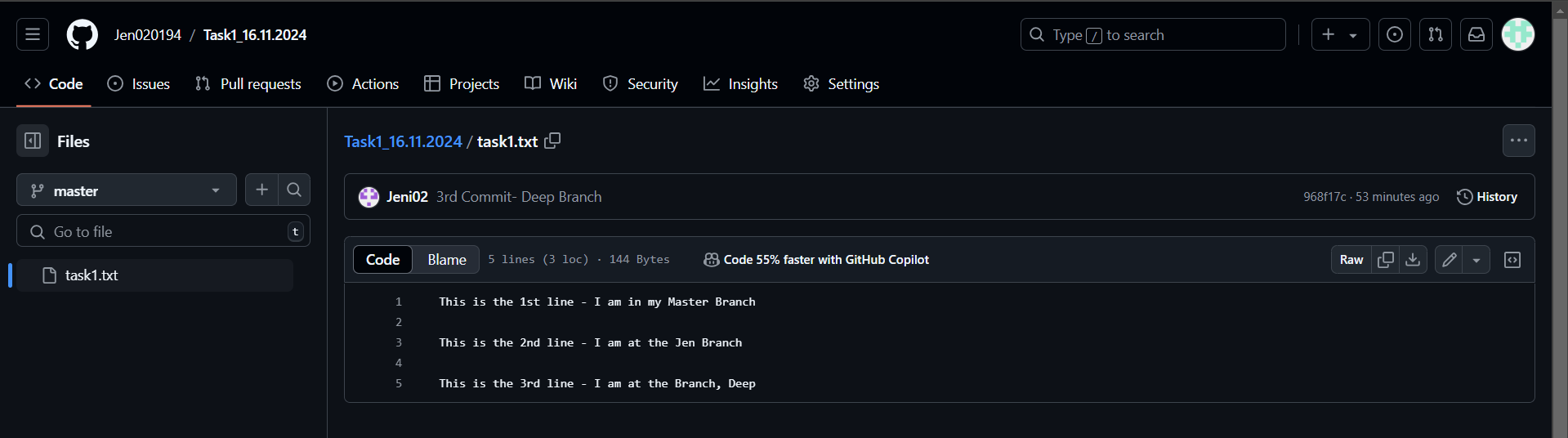
**18. A new Repository created in my GitHub**

****

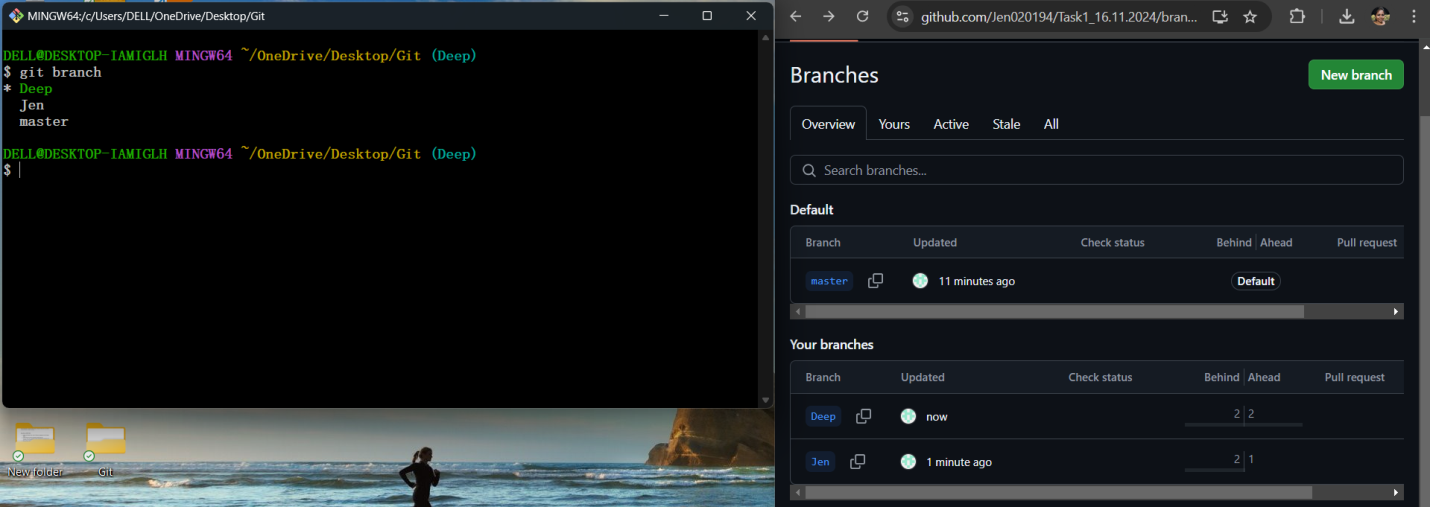
**19. Remote Origin has added using “$ git remote add origin<Repository\_URL>“ command**

****

**19. The file Pushed using “$ git push –set-upstream origin master” command**

****

**24. All the 3 Branches Pushed successfully to the Targeted Repository**

****

**This task covers the most commonly used Git commands.**