A logo for a university

AI-generated content may be incorrect.

AMITY SCHOOL OF ENGINEERING

AND TECHNOLOGY

**SOURCE CODE MANAGEMENT**

LABARATORY RECORD

2nd Semester-2025

FACULTY: Dr. Monit Kapoor

Submitted By:

KISHAN SETTY

B. Tech ES & VLSI

(ASET)

A866181524003

**Lab Exercise 01:** Setting up a Git Environment and Basic Configuration.

**Objective:**

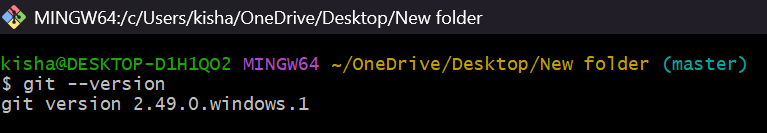
To install Git version control system and configure basic user settings for version control operations.

**Implementation Steps:**

1. Git Installation:
2. Download Git Installer
3. Run the Installer
4. Finish Installation
5. Verify Installation
6. Installation Verification:

Command: git --version

Output:



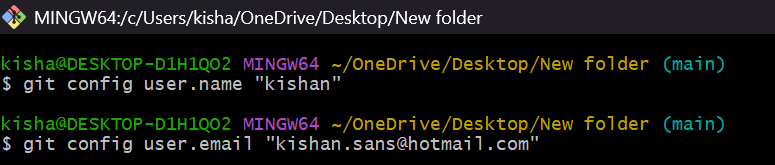
1. Git configuration:

commands: $ git config user.name "kishan"

$ git config user.email "Kishan.sans@hotmail.com"

$ git config user.name

Output:



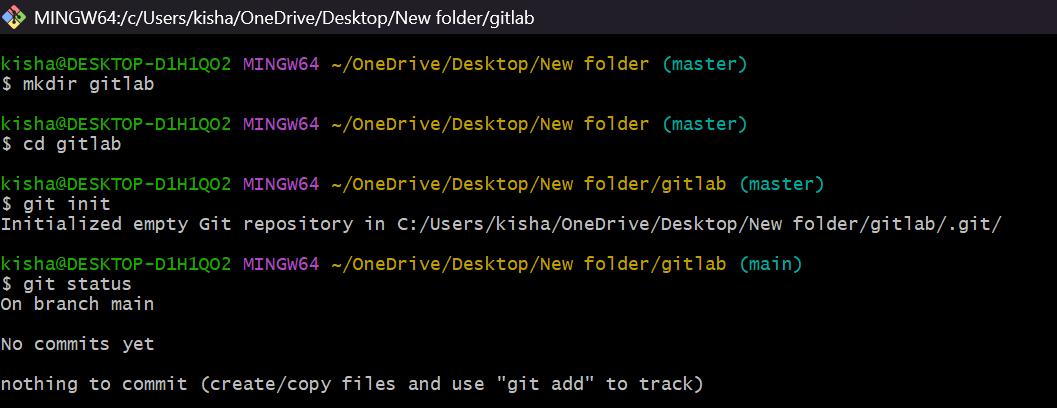
1. Project Setup:

Commands: mkdir gitlab

cd gitlab

git init

Output:

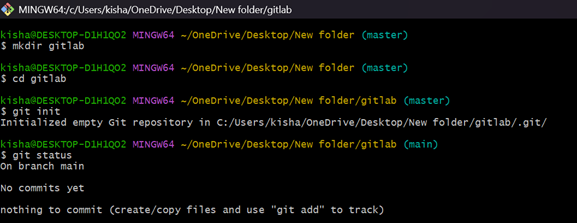


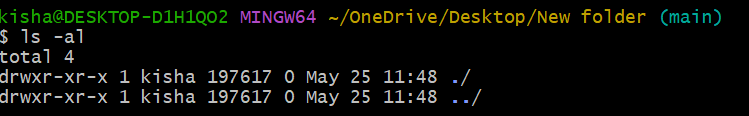
**Lab Exercise 2:** Creating and Managing Local Repositories

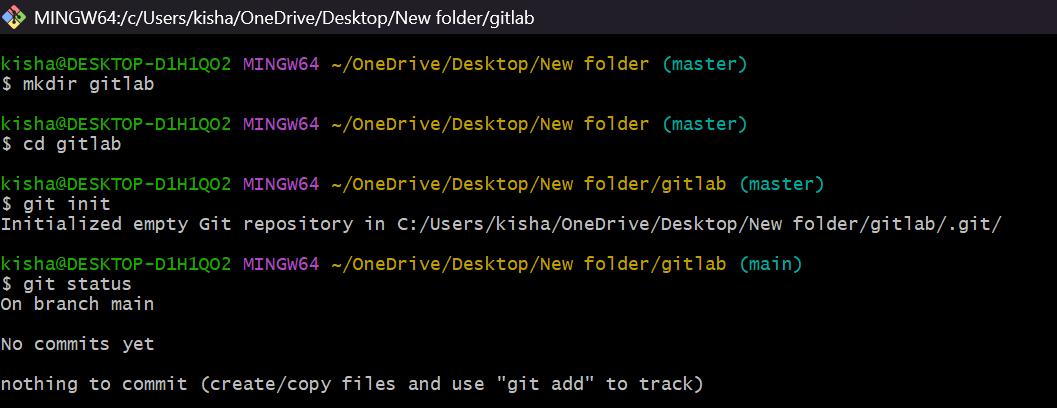
**Objective:**  To create a local Git repository and understand its basic structure and initialization process.

**Implementation Steps:**  
  1. Repository Creation:  
  # Commands: mkdir gitlab

cd gitlab

git init  
  # Output  
   

  2. Repository Structure Analysis:  
      Commands : ls -al  
      Output  
   

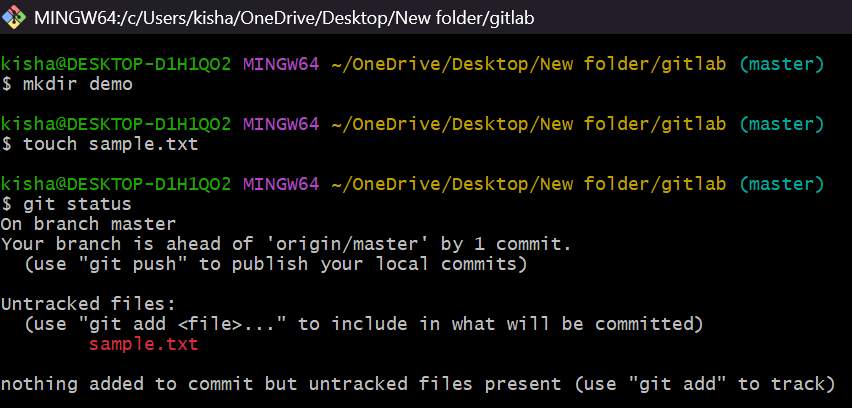
3. Repository Status Check:  
     #Commands: git status  
      Expected Output  
    

**Lab Exercise 03:** Working with Basic Git Commands (add, commit, status)

1. Creating Sample File:

Commands: touch sample.txt

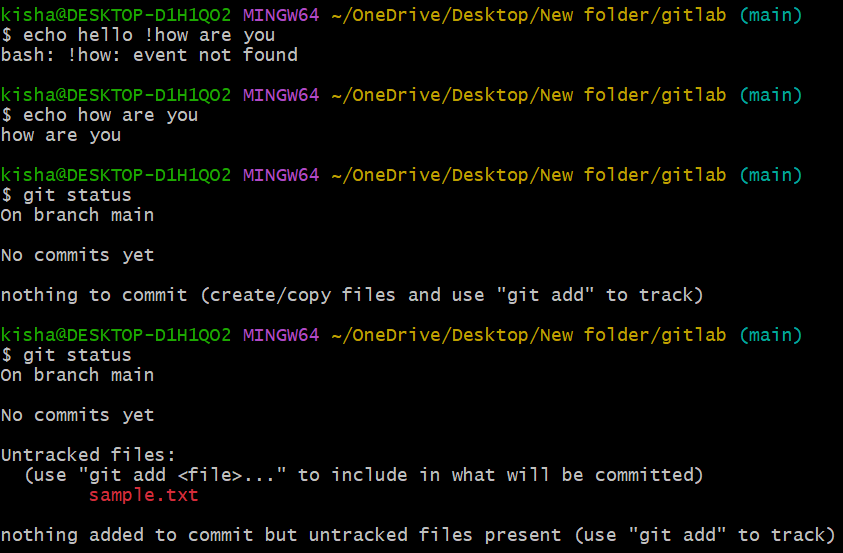
Output:



2. Checking Repository Status:

Commands: git status

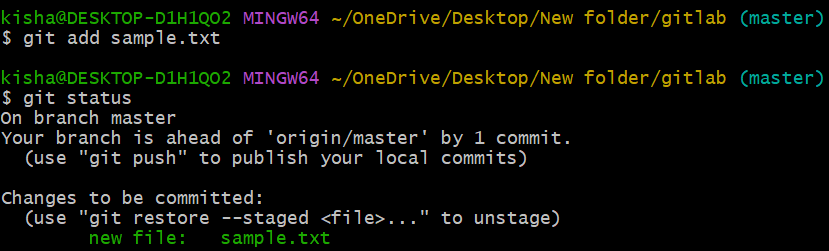
Output:



3. Adding File to Staging Area:

Commands: git add sample.txt

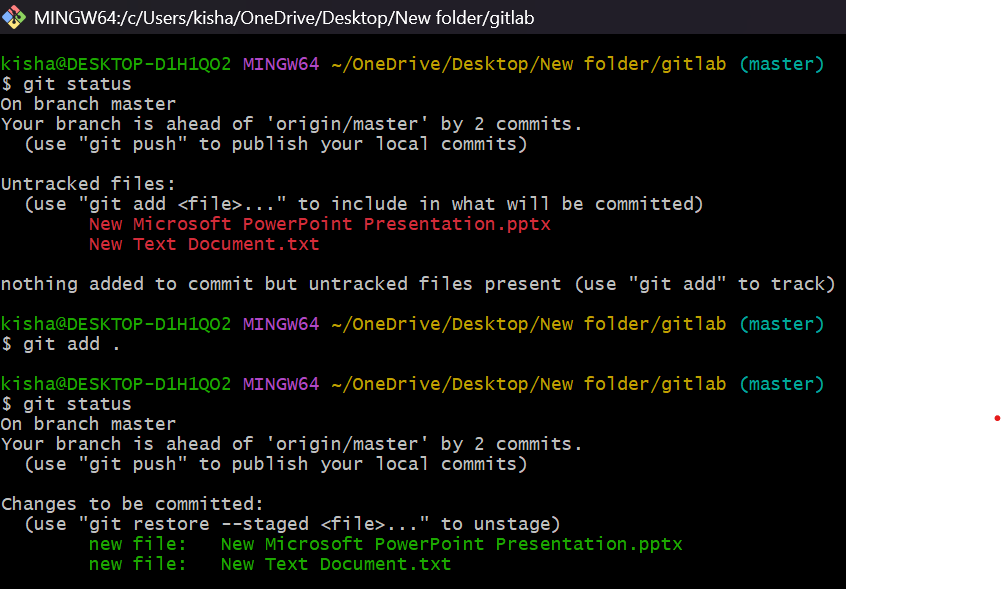
Output:



4. Adding Multiple Files:

Commands: git add .

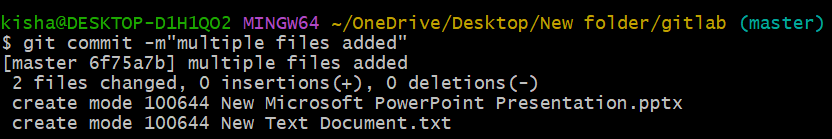
Output:



5. Creating Initial Commit:

Commands: git commit -m “multiple files added”

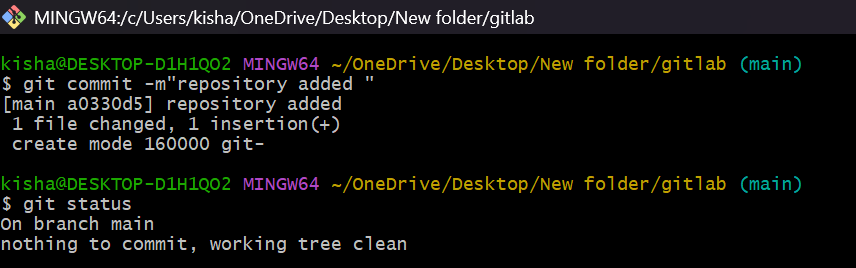
Output:



6. Modifying Files:

# Commands: echo “Adding new content” >> sample.txt

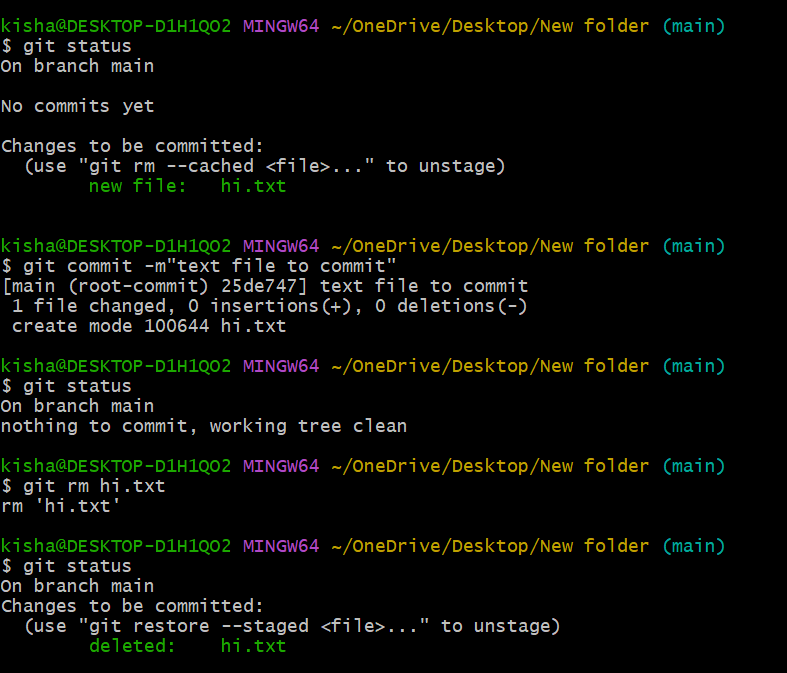
# Output:



7. Viewing Changes:

# Commands: git diff sample.txt

# Output:

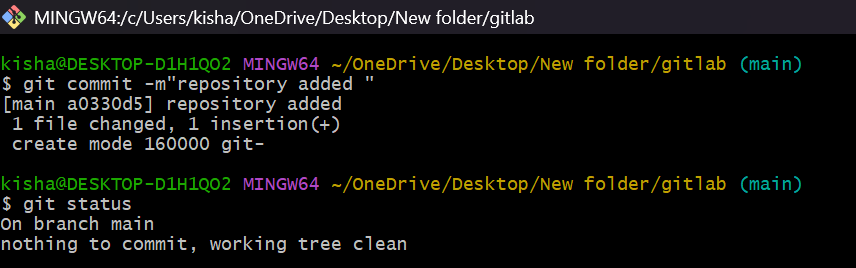


8. Adding and Committing modified files:

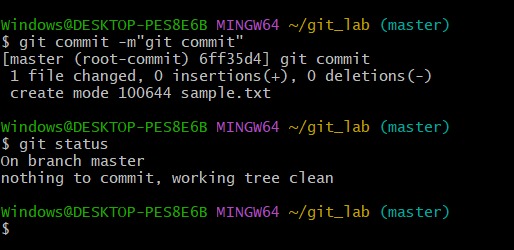
# Commands: git add sample.txt

git commit -m “added text file”

# Output:

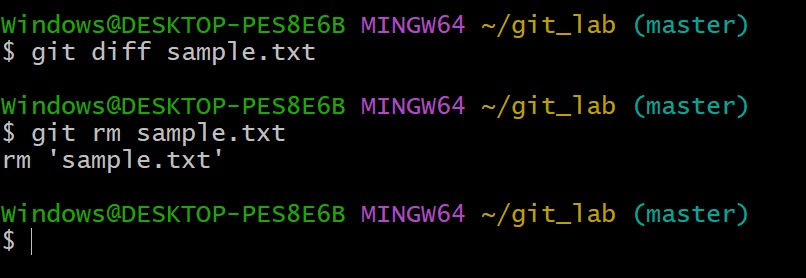


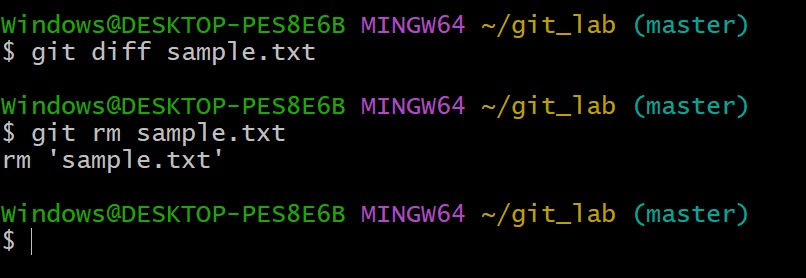
**Objective:**  To understand and practice Git commands for managing files and directories, including adding, tracking modifications, removing, and renaming files.

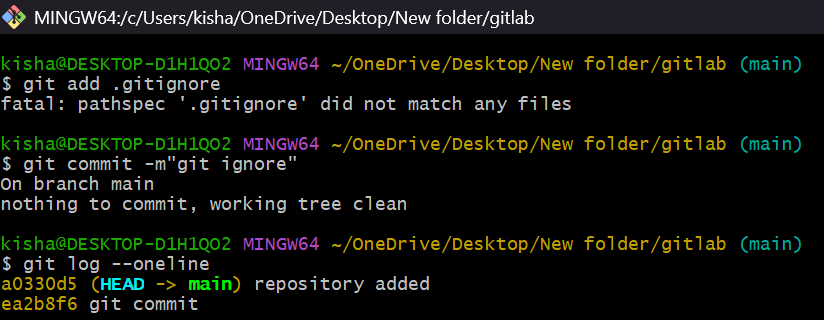
**Implementation Steps:**  
  1. Adding New Files to Repository:  
   
  # Output:  


  2. Tracking Modifications:  
  # Commands: git diff file01.txt

git add file01.txt  
  # Output:

  3. Adding All Changes at Once:  
  # Commands: git add .  
  # Output  


 4. Removing Files from Repository:  
  # Commands: git rm file03.txt  
  # Output  


  5. Ignoring Files:  
   Commands: git .ignore  
   Output  


Lab Exercise 6: Creating and Switching Branches

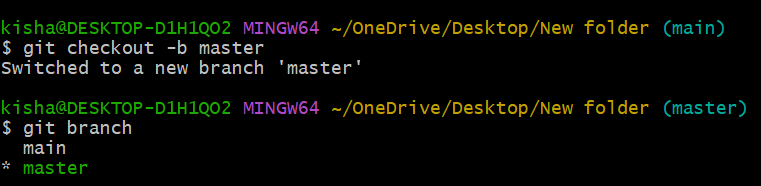
Objective:   
To understand and practice Git's branching system, including creating branches, switching between branches, and managing branch operations.

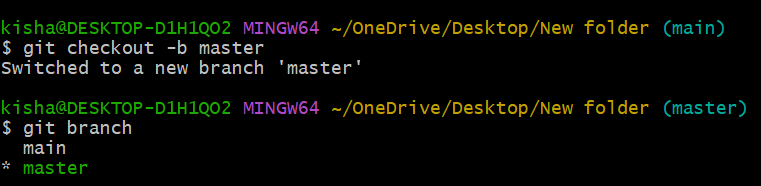
  1. Branch Creation:  
  #Commands

# Create new branch: git branch main

# Create and switch to a new branch in one command: git checkout -b master

#Output:





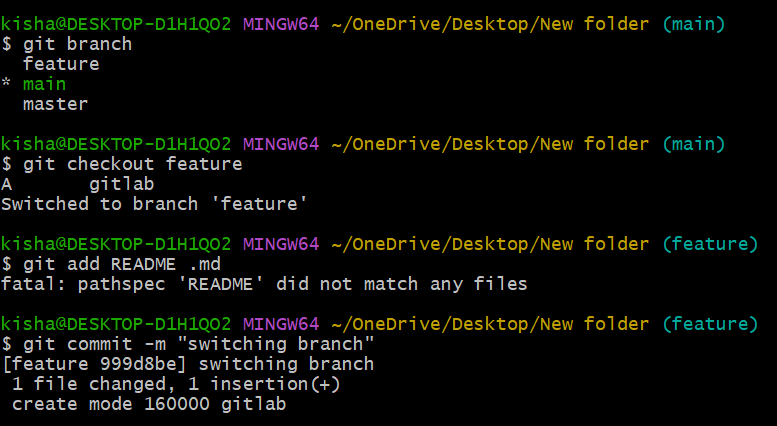
  2. Switching Between Branches:  
  # Switch back to main branch: git checkout master

  # View all branches in repository: git branch

   # Switch to another branch: git checkout feature

   # Use newer Git switch command: git switch feature

#Output:



Merging branches and resolving conflicts

Lab Exercise 7: Merging branches and resolving conflicts

Objective:  
To understand and practice two fundamental Git merge strategies: Fast-Forward Merge and Three-Way Merge.

Prerequisites:  
- Git installed  
- Basic understanding of Git branches  
- Command line interface familiarity

Environment Setup:

1. Fast-Forward Merge Demonstration:

Scenario:  
Create a repository and demonstrate a simple, linear merge where no additional changes exist in the main branch.

 #Commands: nano README1.md

git add README1.md

git commit -m”1234 to README1.md”

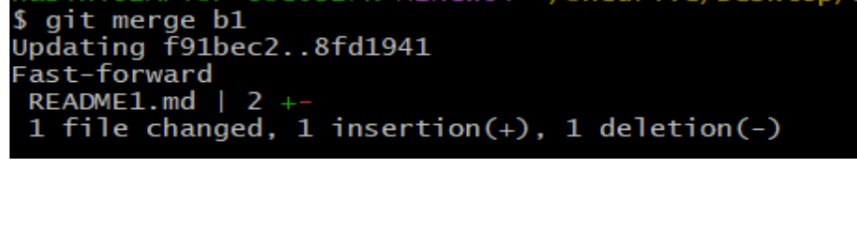
git checkout -b b1

nano README1.md

git add README1.md

git commit -m”added 56 in README1.md”

git checkout master

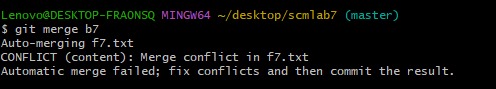
  
#Output

 2. Three-Way Merge Demonstration:

 Scenario:  
Create a scenario where both main and feature branches have different commits.

 #Commands: nano f7.txt

git add f7.txt

 git commit -m “1234 in f7.txt”

git checkout master

git checkout -b b7

nano f7.txt

git add f7.txt

git commit -m “9101112 is added to f7.txt”

    git merge b7  
#Output:

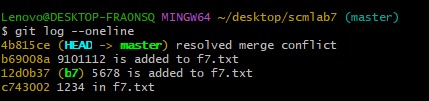
Potential Merge Conflict:

Resolving Conflicts: nano f7.txt

git add f7.txt

git commit -m “resolved merge conflict”

Output:



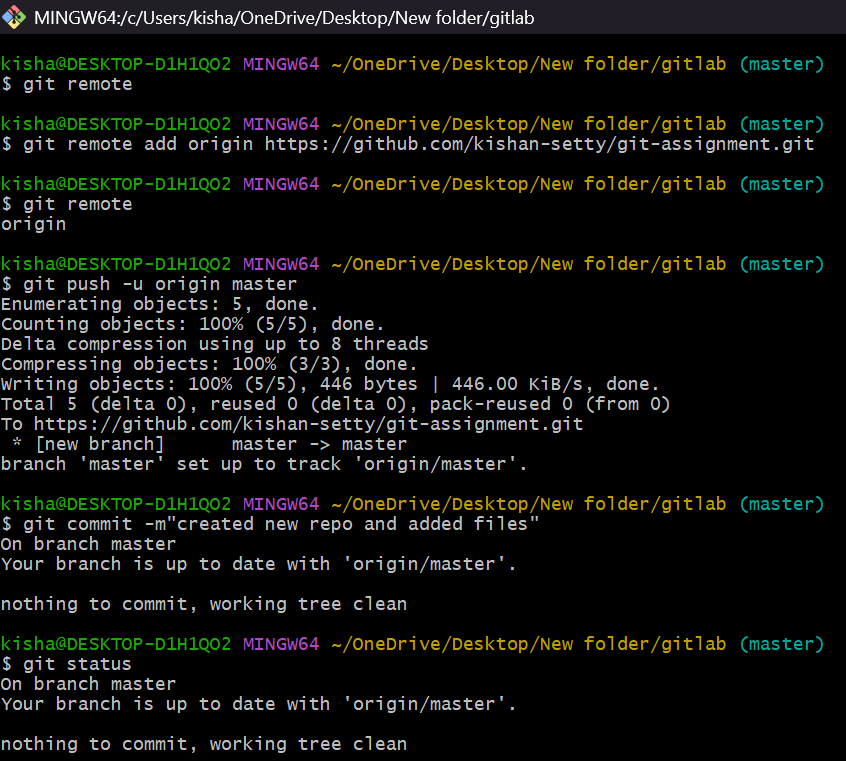
Lab Exercise 8: Working with Remote Repositories

Objective  
To understand and practice working with remote Git repositories, including connecting to remote repositories, pushing local changes, and pulling remote changes.

Prerequisites  
- Git installed  
- Basic understanding of Git branches and commits  
- Command line interface familiarity  
- GitHub/GitLab/Bitbucket account (or any Git hosting service)

Case 1: Pushing a Local Repository to a Remote Repository

#Output:



Case 2: Cloning and Pulling from a Remote Repository

Scenario: A remote repository exists that you want to work with locally. You'll clone it, observe changes made to the remote repository, and pull those changes to your local copy.

#Output: 