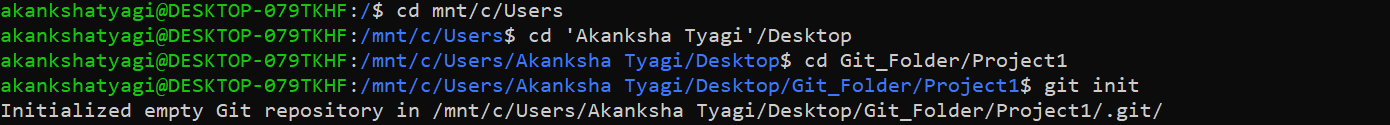
II. Version Control

Akanksha Tyagi

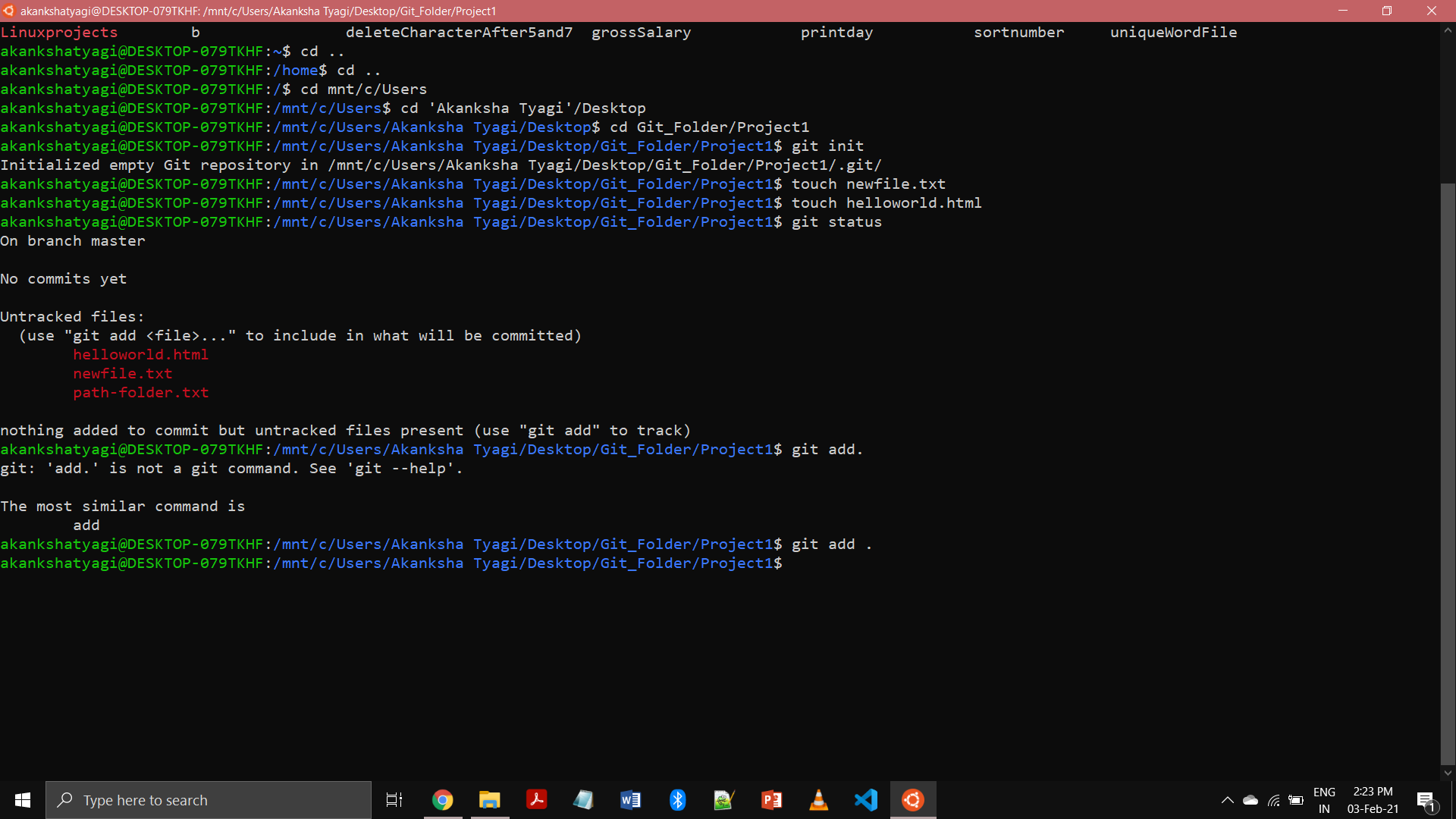
Id- 4701

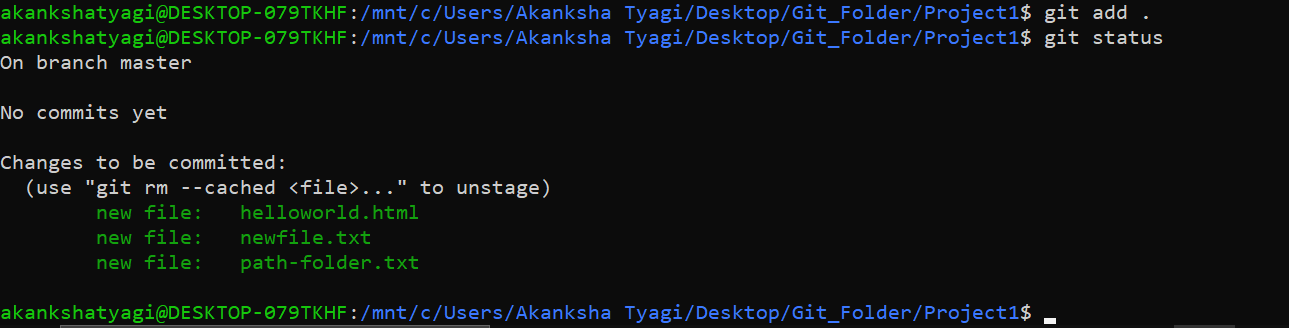
JVM Trainee

1. Git Setup
2. 2. Initialize a Git Repository

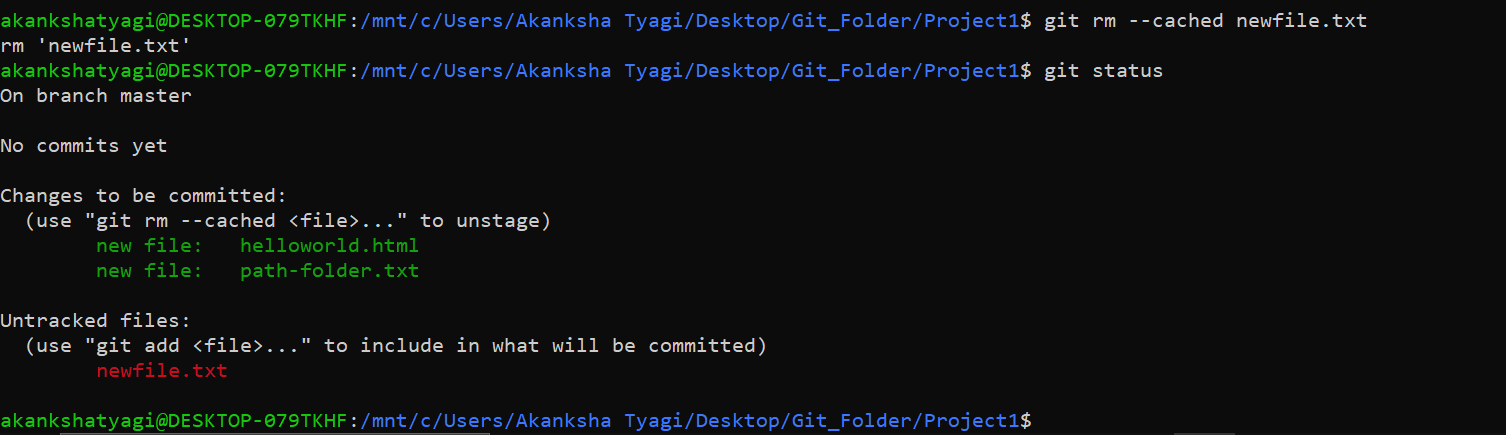


3. Add files to the repository

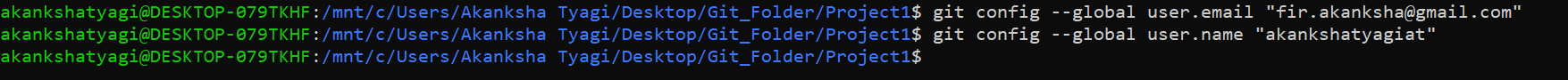


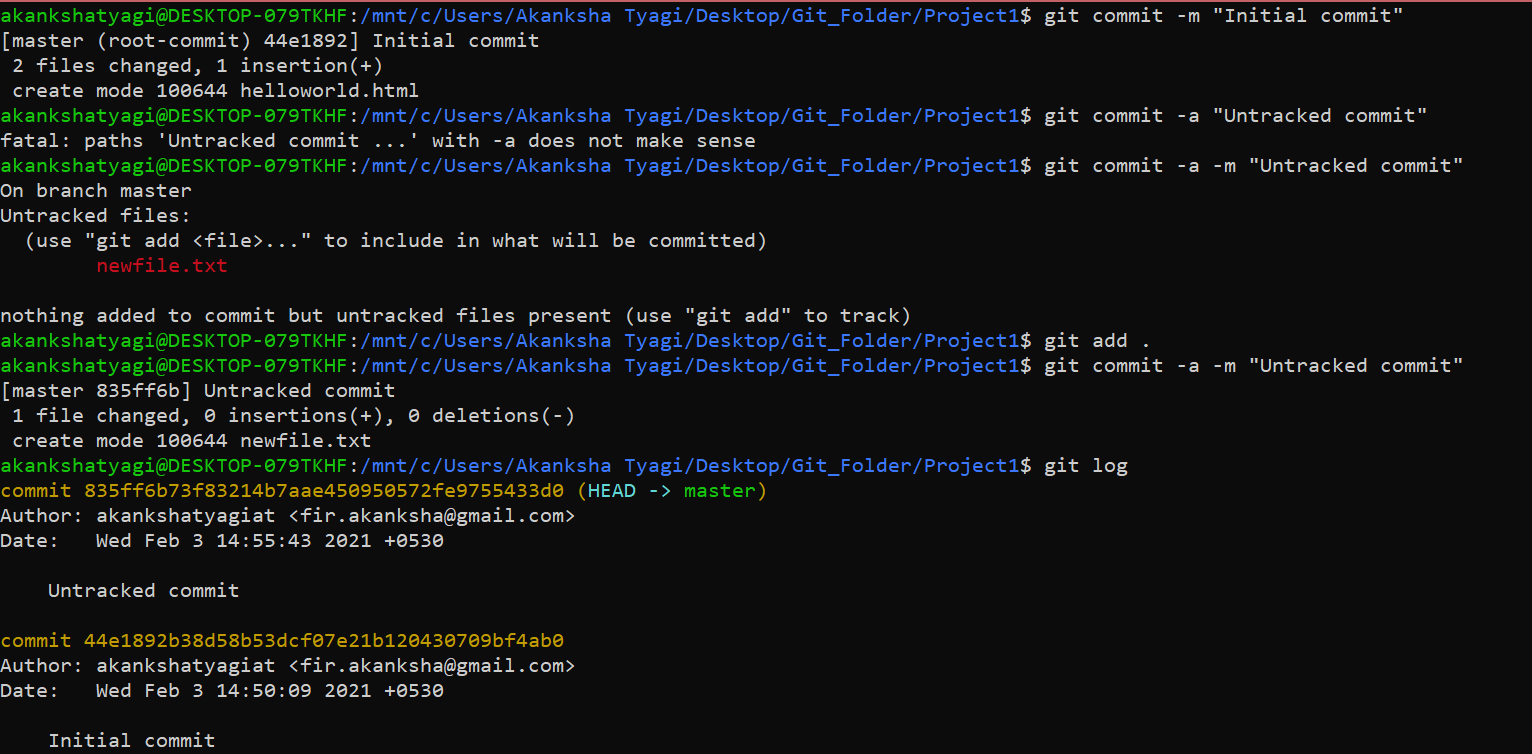


4. Unstage 1 file



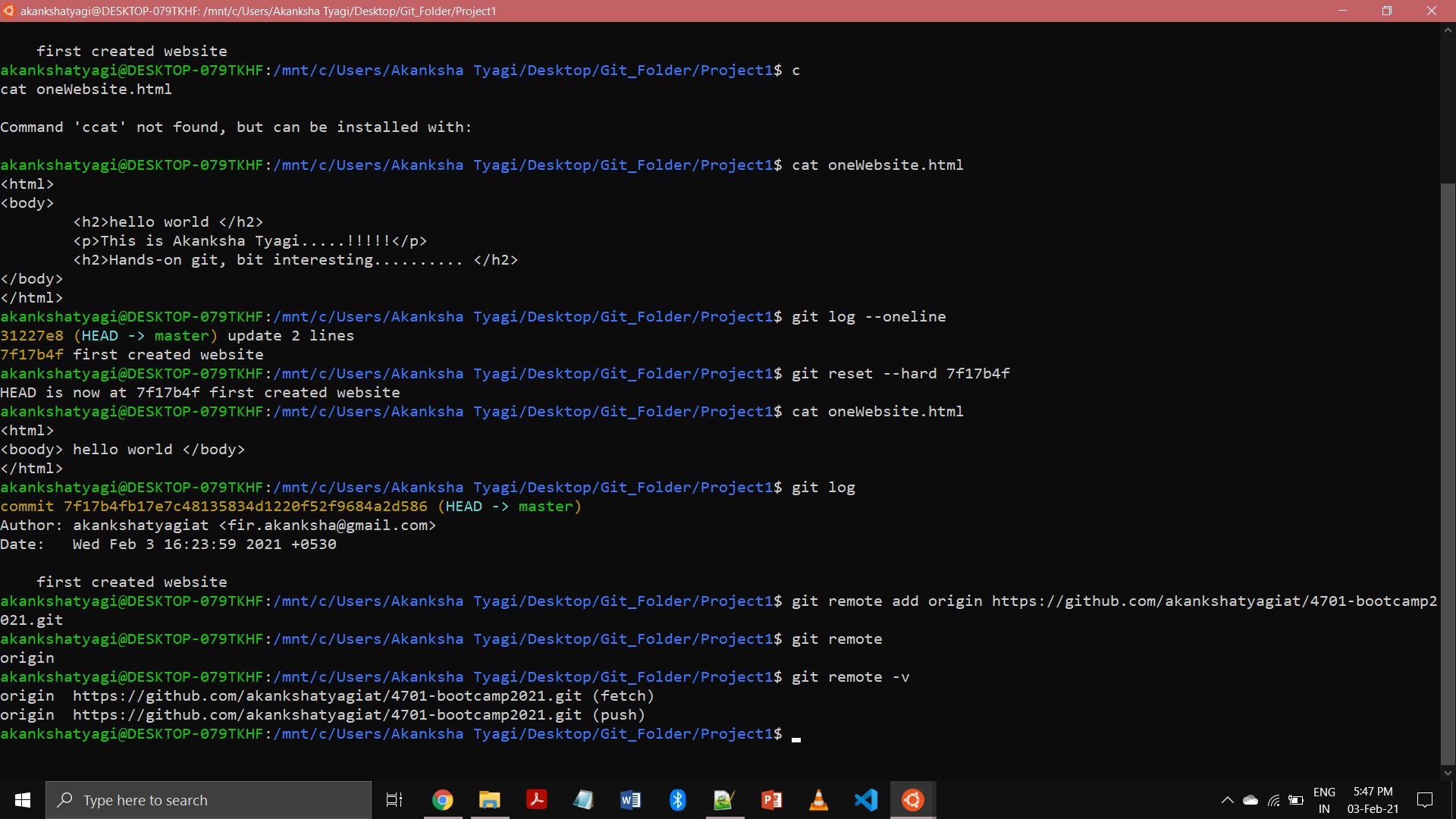
5. Commit the file





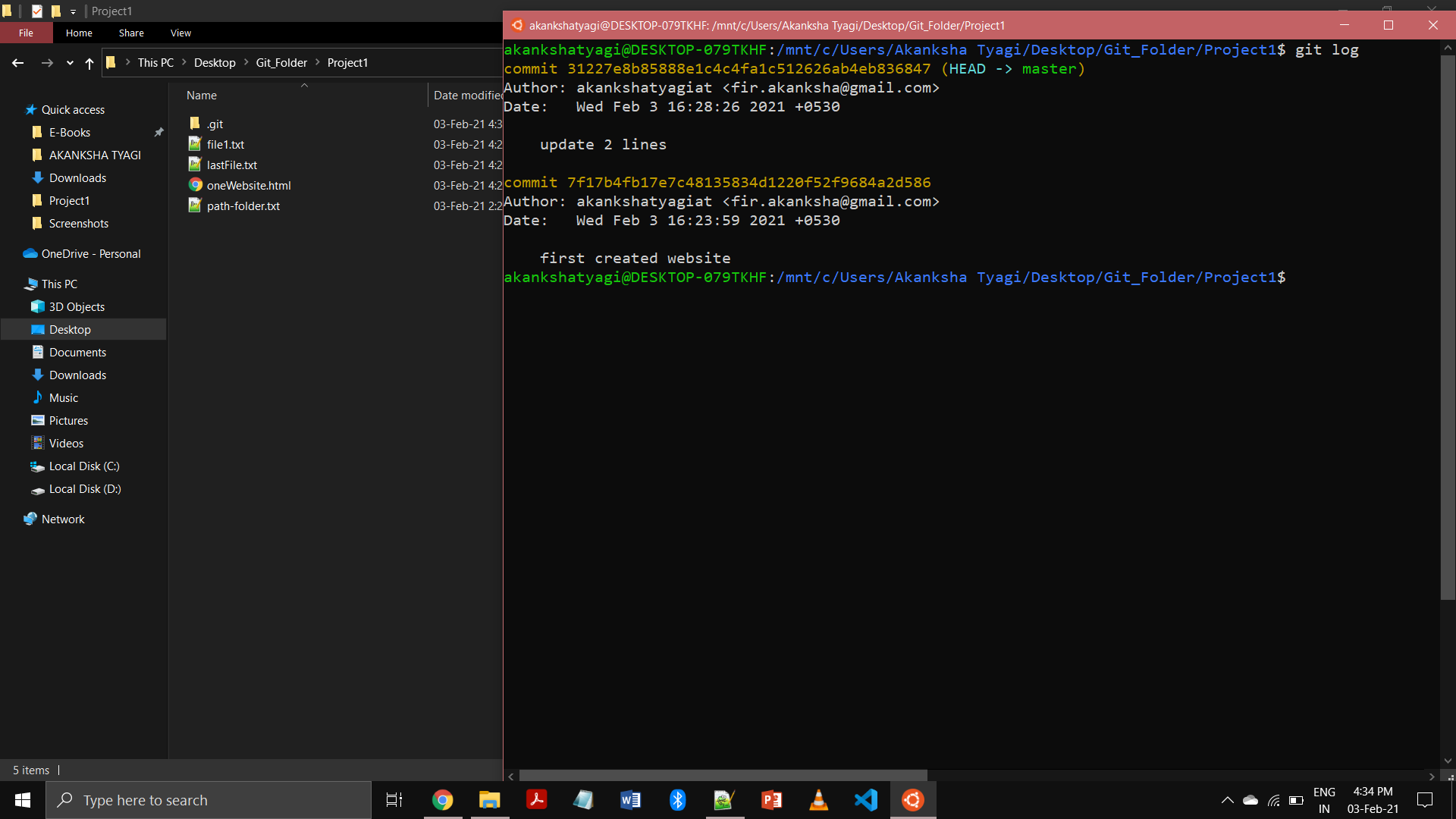
6. Add remote

=> git remote add origin <url>

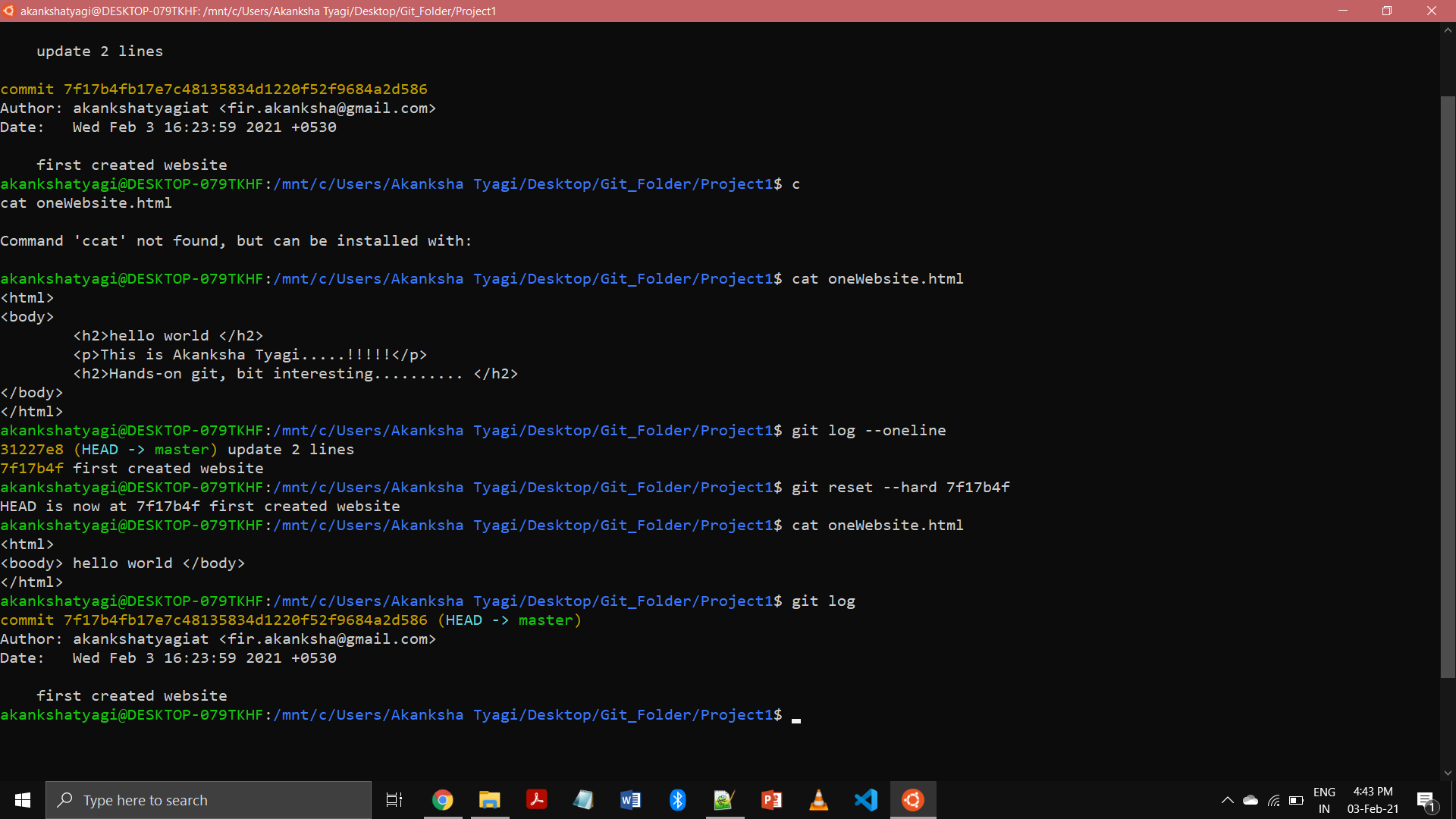


7. Undo changes to a particular file

Before reverting back changes:

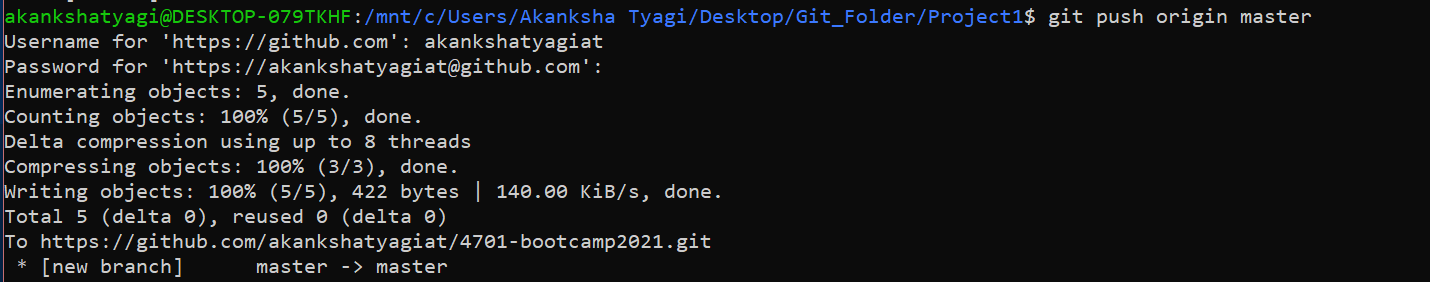


After reverting



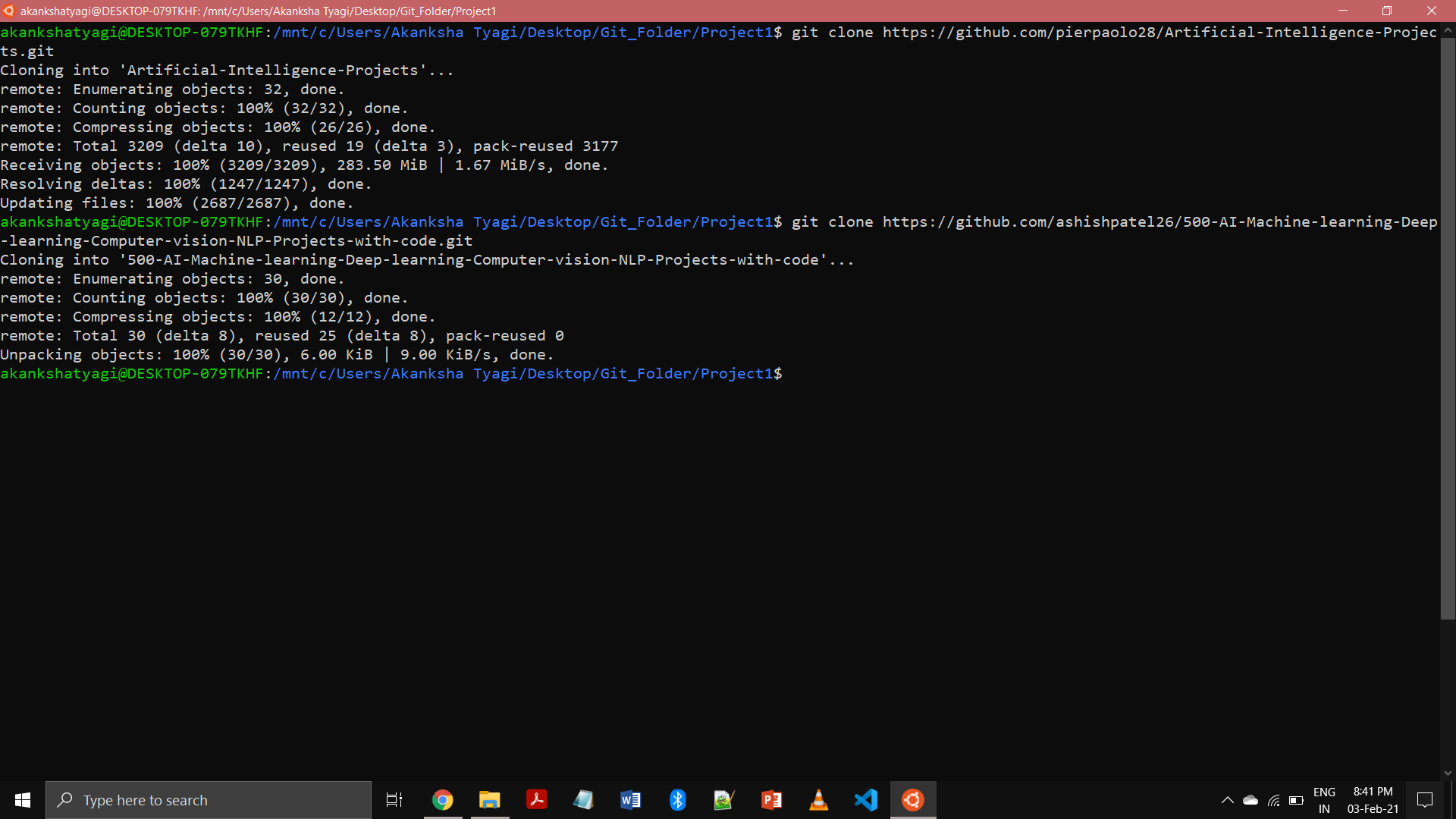
8. Push changes to GitHub

=> git push origin master



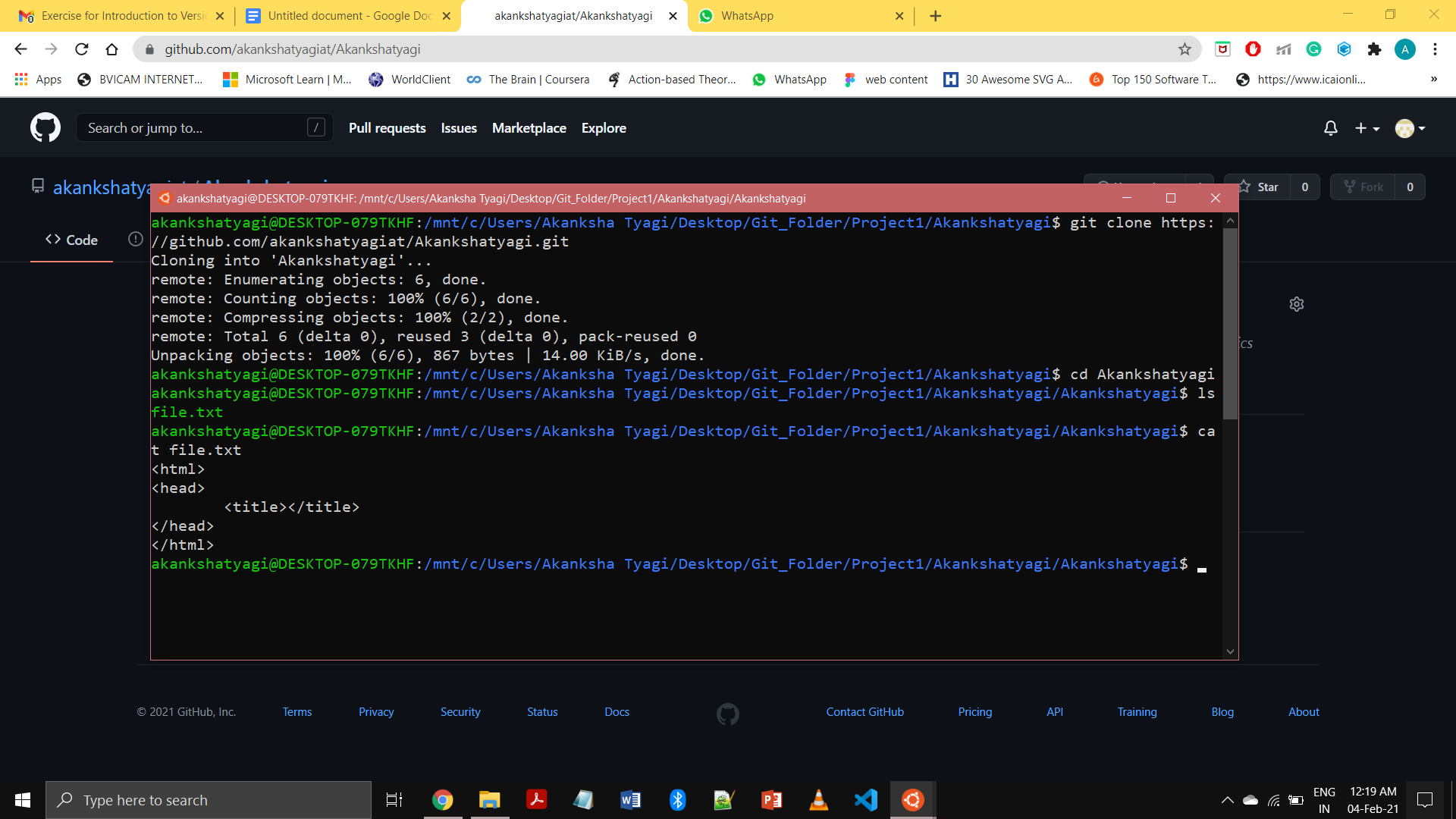
9. Clone the repository

=> git clone <url>

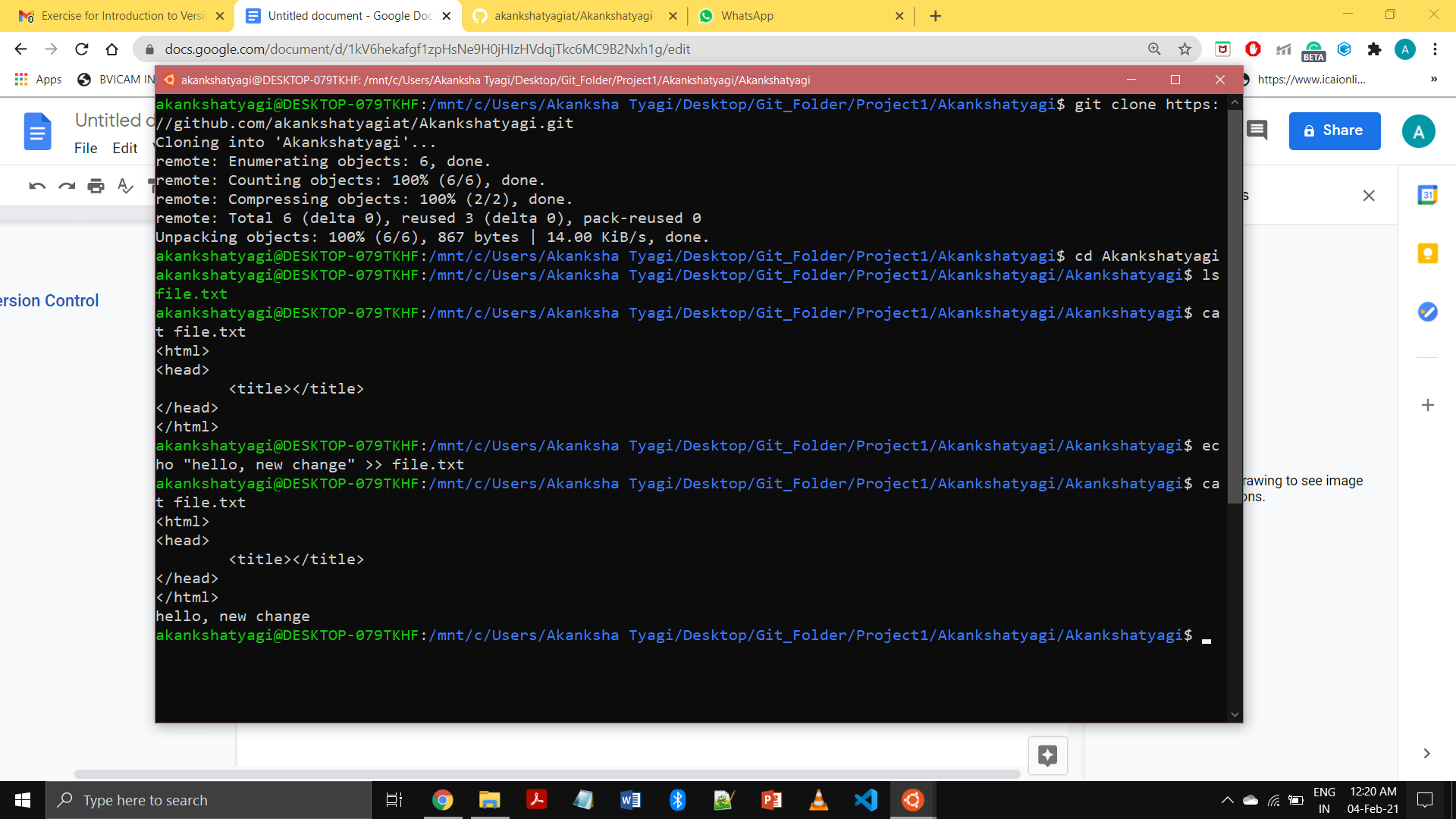


10. Add changes to one of the copies and pull the changes in the other.

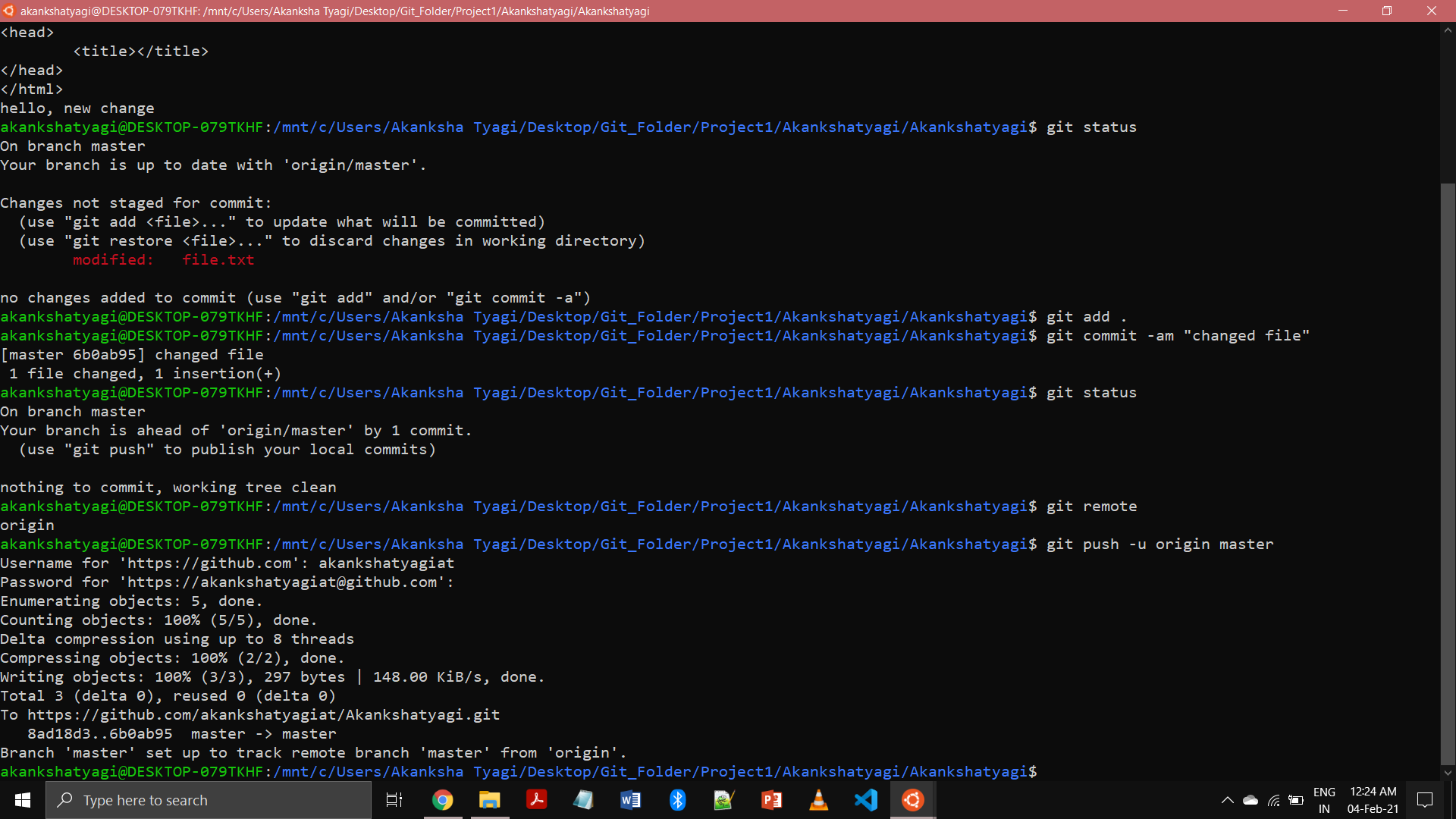
Cloned a repository



and changing contents of “file.txt”

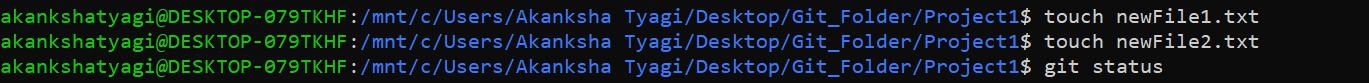


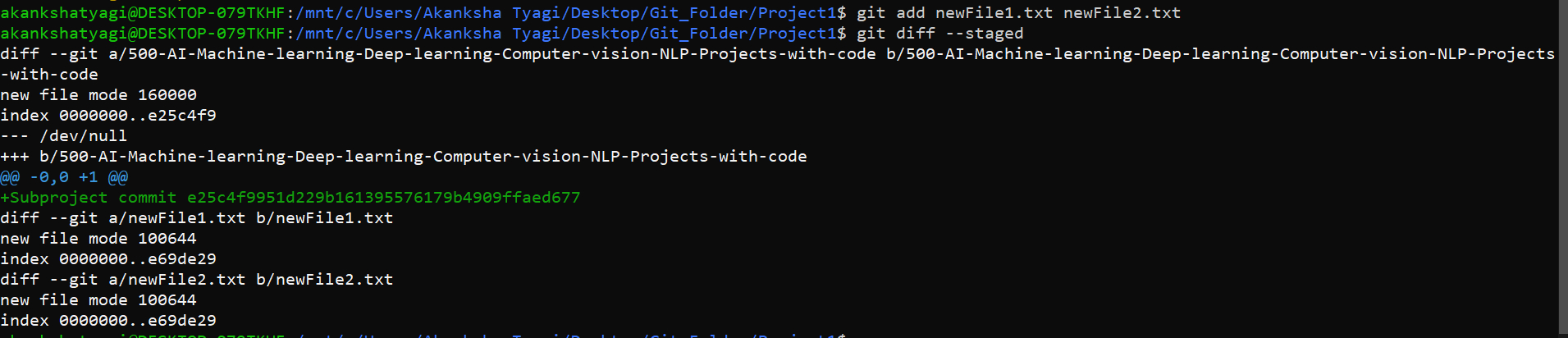
Commiting the changes in the file “file.txt” and lastly pushing back the repository on github.



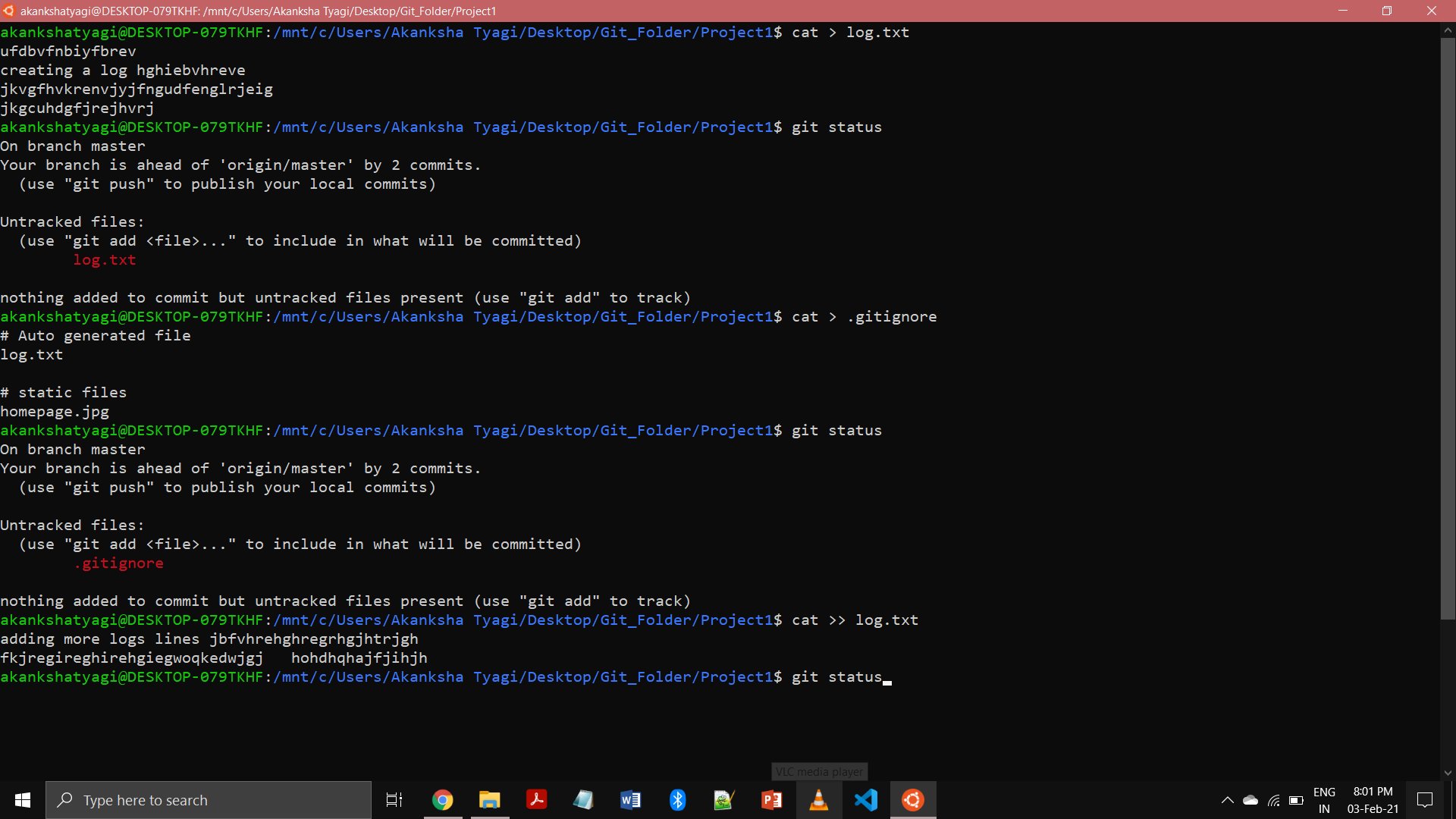
11. Check differences between a file and its staged version

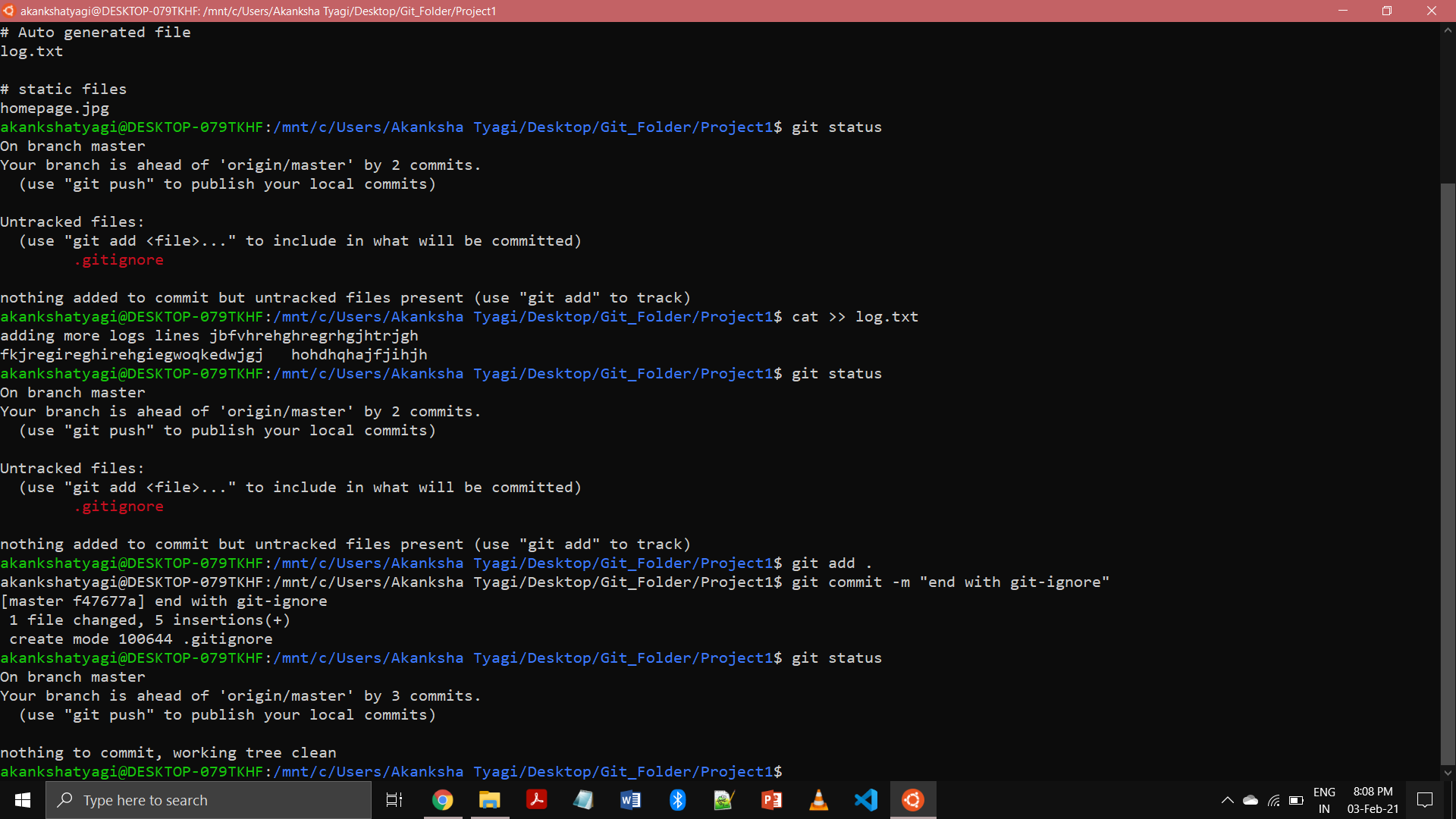
⇒ git diff --staged



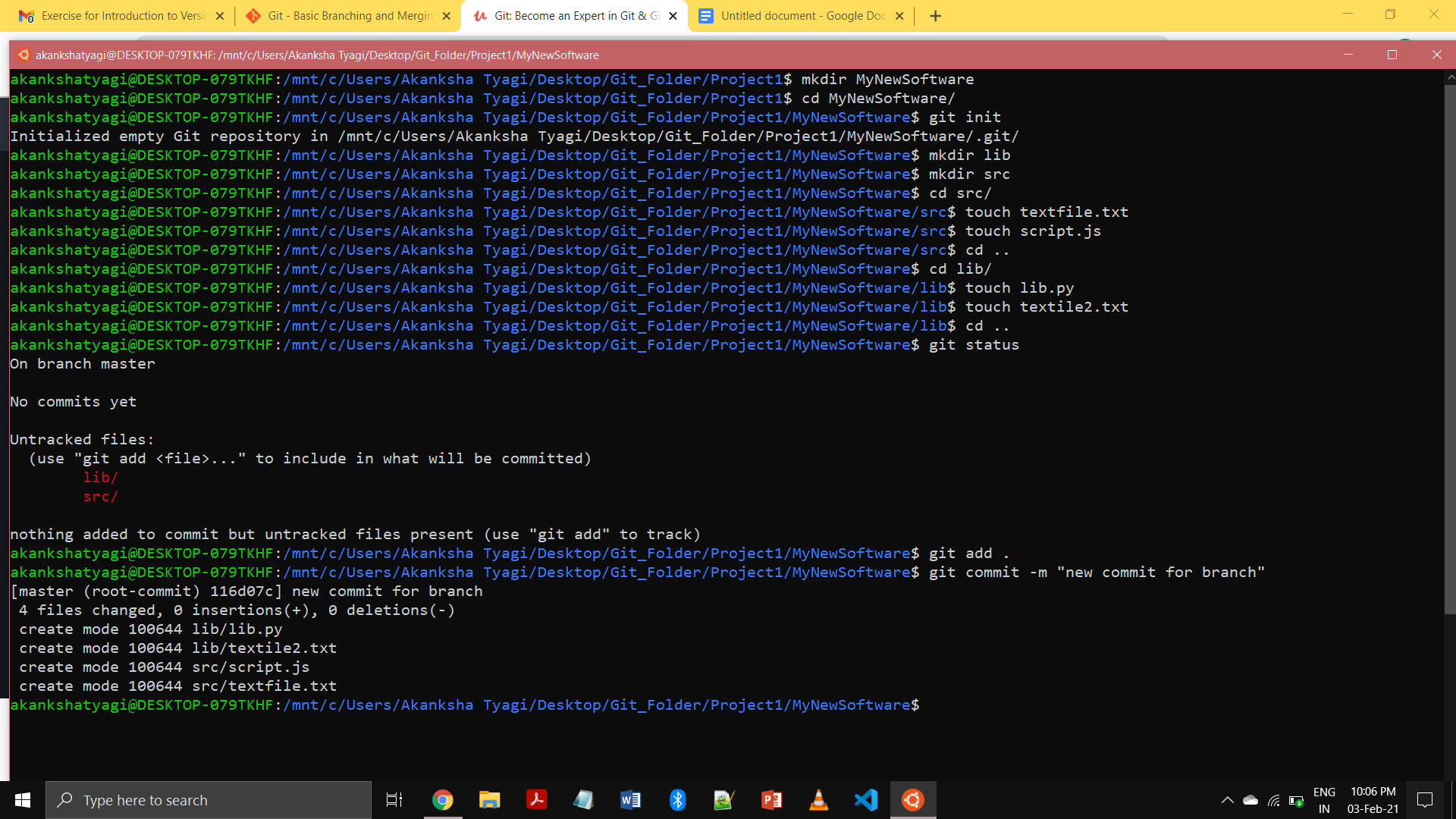


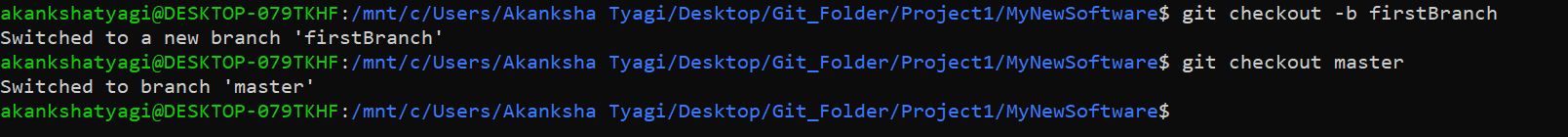
12. Ignore a few files to be checked in





13. Create a new branch.

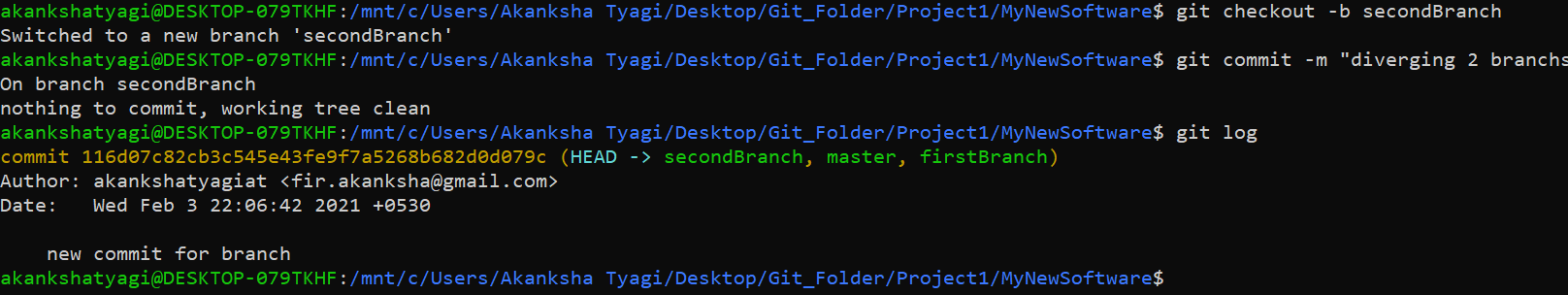




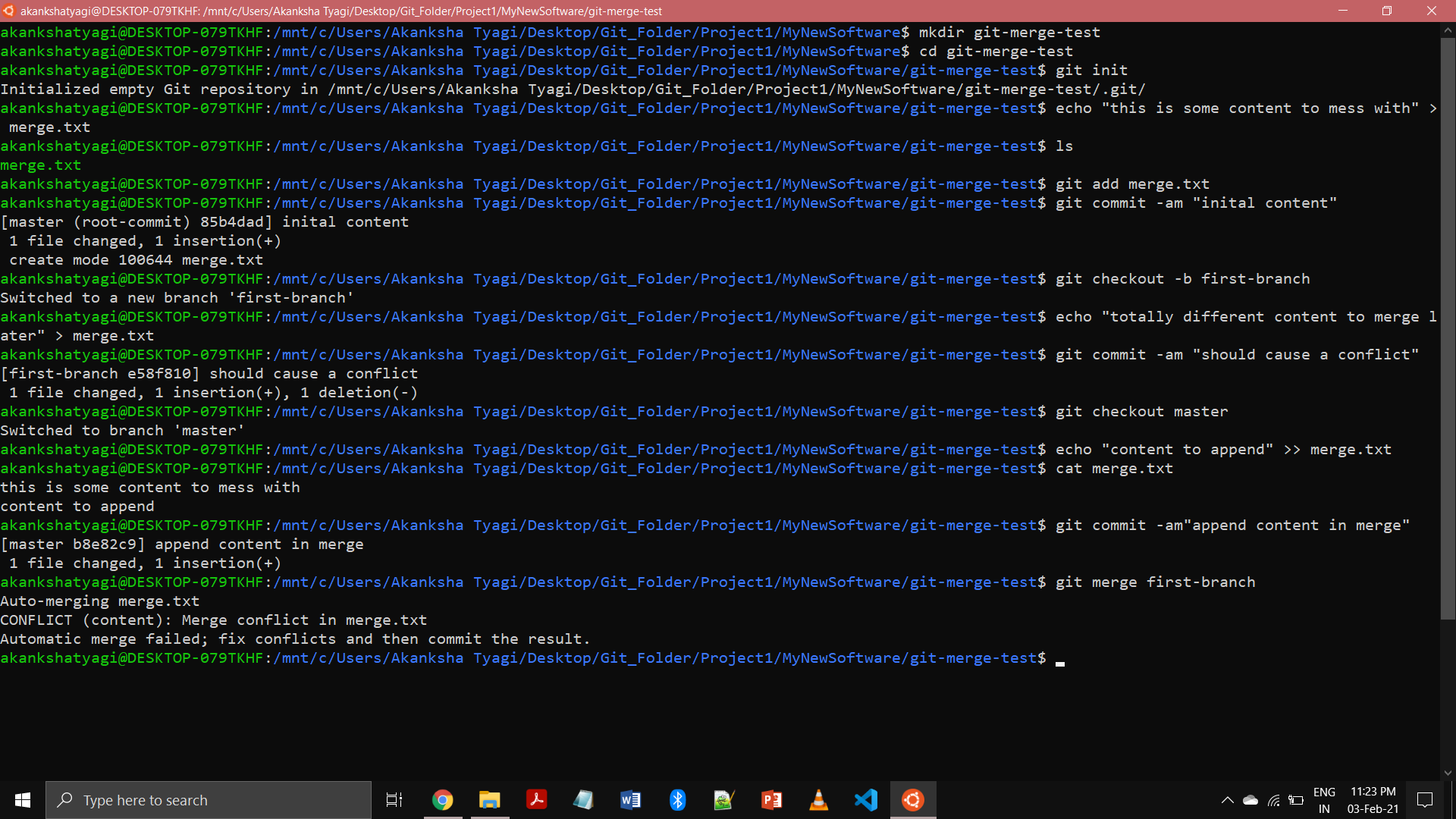
14. Diverge them with commits

=> git checkout -b <branch-name>

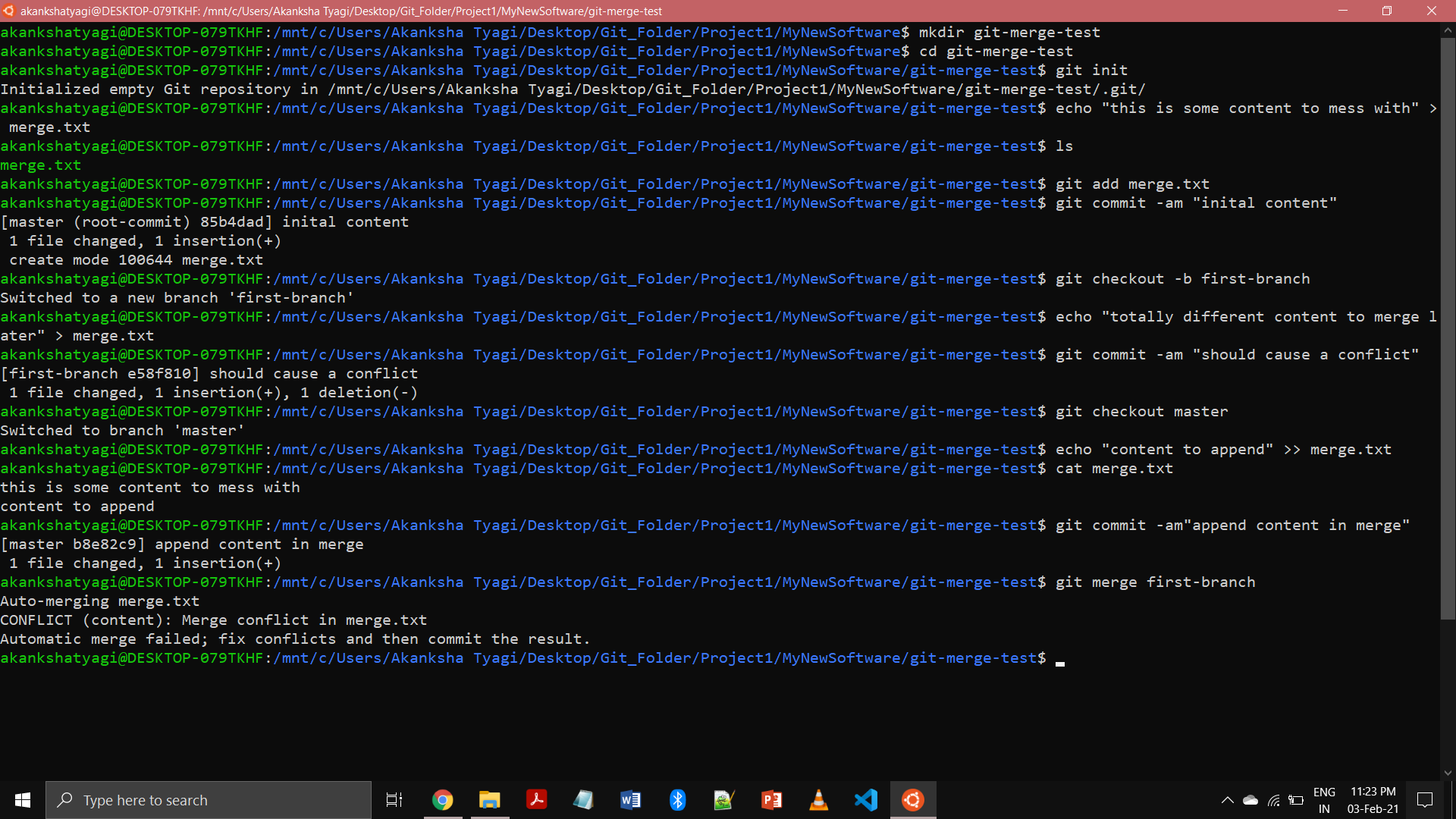
=> git commit -m “diverging 2 branches”

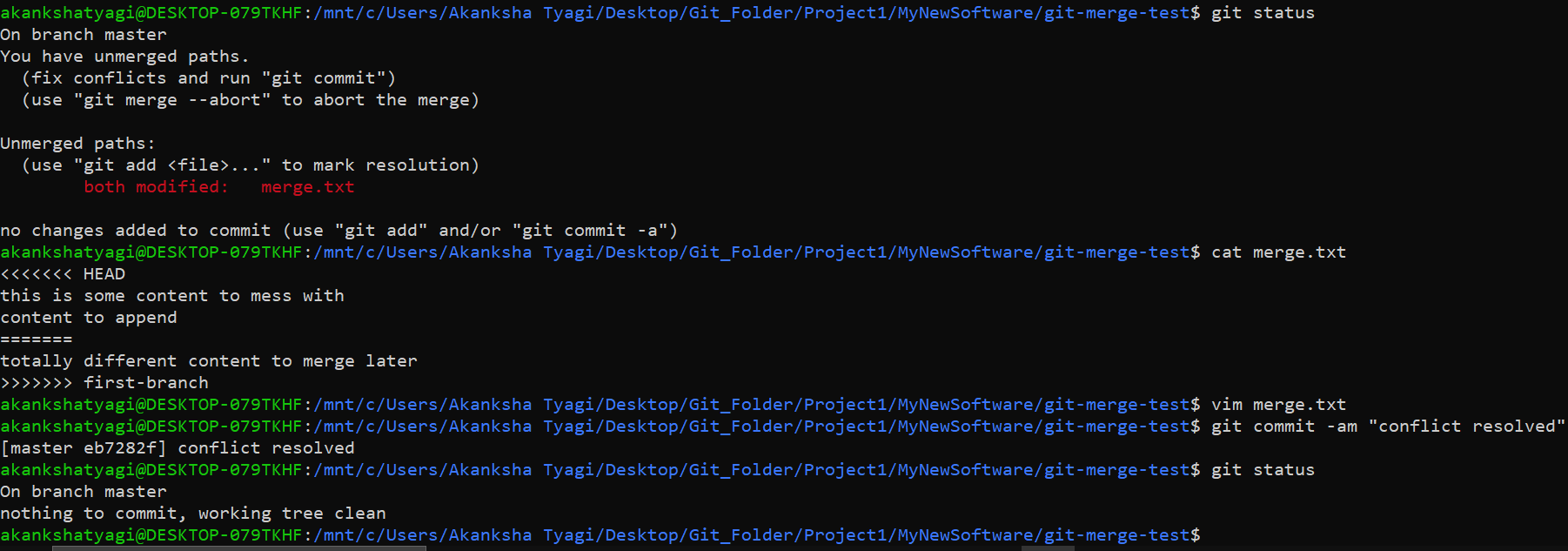


15. Edit the same file at the same line on both branches and commit



16. Try merging and resolve merge conflicts





17. Stash the changes and pop them



18. Add the following code to your .bashrc file : color\_prompt="yes"  
parse\_git\_branch() {  
git branch 2> /dev/null | sed -e '/^[^\*]/d' -e 's/\* \(.\*\)/(\1)/'  
}  
if [ "$color\_prompt" = yes ]; then  
PS1='\u@\h\[\033[00m\]:\[\033[01;34m\]\W\[\033[01;31m\] $(parse\_git\_branch)\[\033[00m\]\$ '  
else  
PS1='\u@\h:\W $(parse\_git\_branch)\$ '  
fi  
unset color\_prompt force\_color\_prompt

