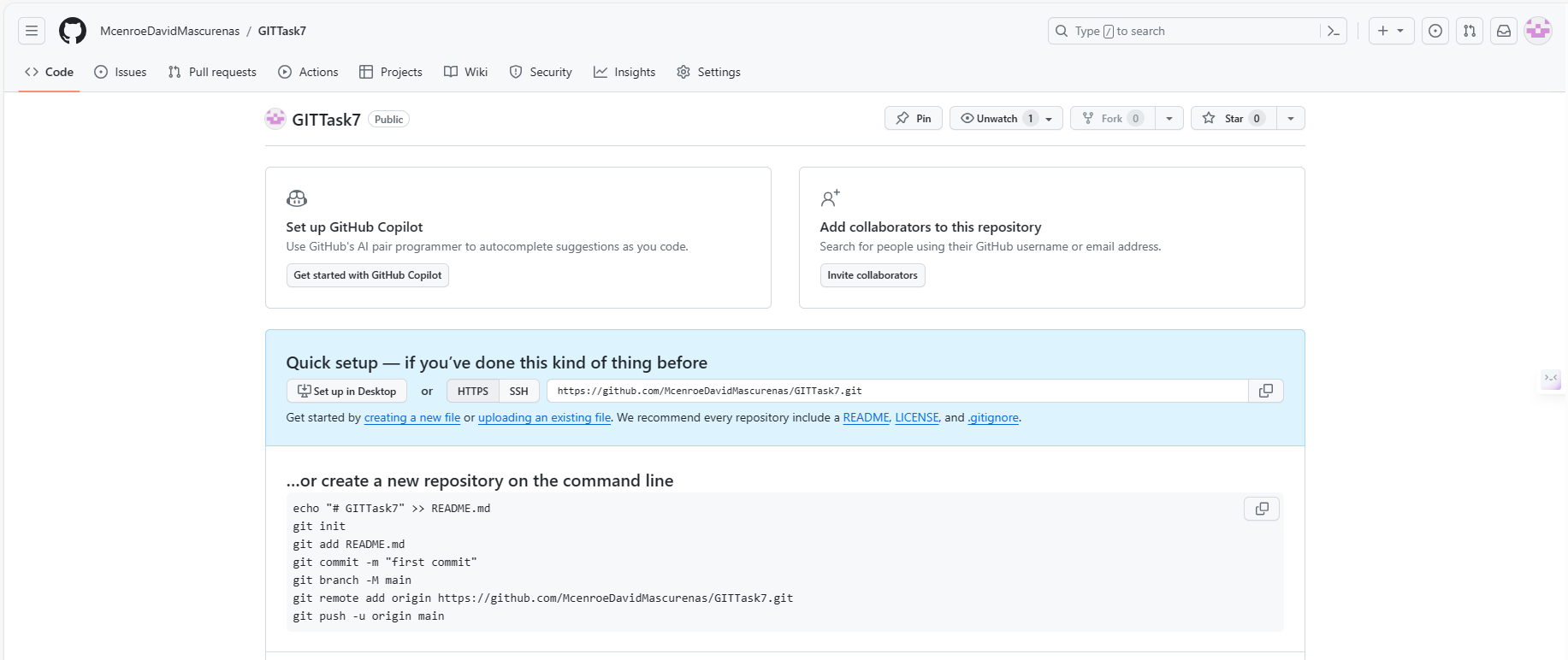
Git & Github

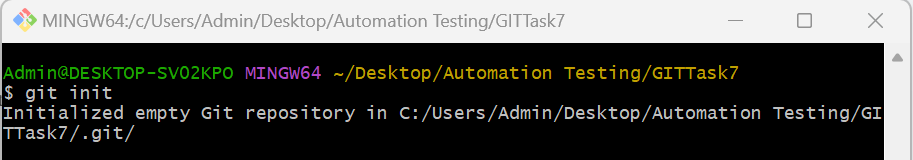
1. Create a New repository in github.

* Navigate to GitHub: Go to the GitHub website (<https://github.com/>) and make sure you are logged in.
* Click "New repository": In the upper right corner of any page, click the "+" icon and select "New repository".
* Fill in the details:
  + Name: Enter a short, descriptive name for your repository. This is important as it will be used to identify your project.
  + Description (optional): Add a brief description of your project. This helps others understand what your repository is about.
  + Visibility: Choose whether you want your repository to be public (visible to everyone) or private (only visible to you and collaborators).
  + Initialization options: You can choose to initialise your repository with a README file, a licence file, or even a template repository. This depends on your project needs.

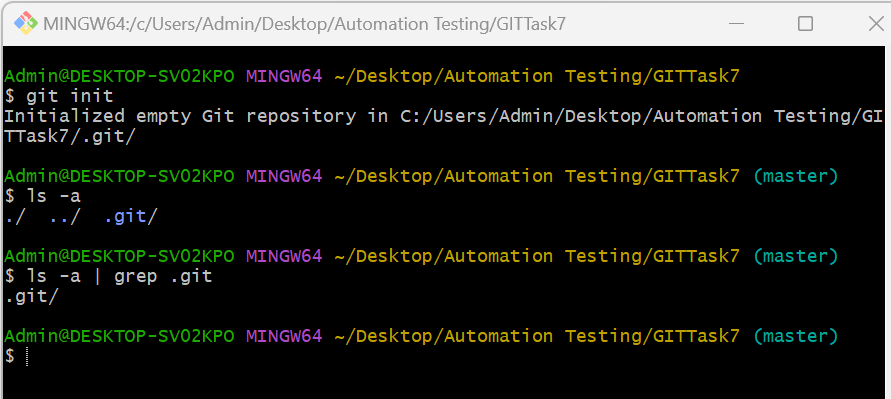
Click "Create repository": Enter the details, click the "Create the repository” button.



1. Use the init command to create a git repository in that directory.

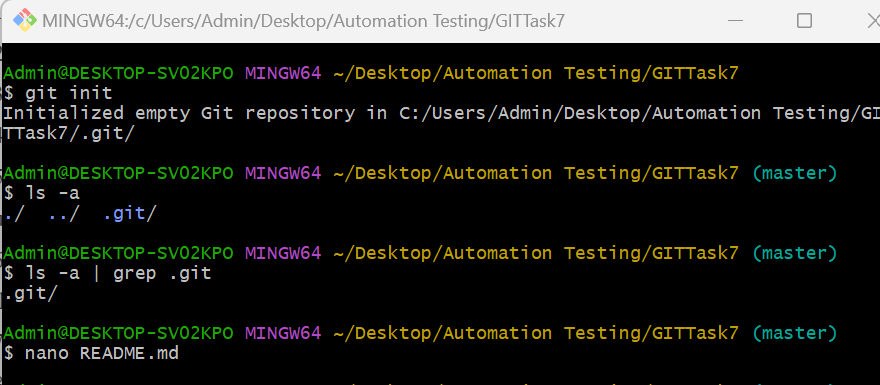


1. Observe that there is now a GIT Directory



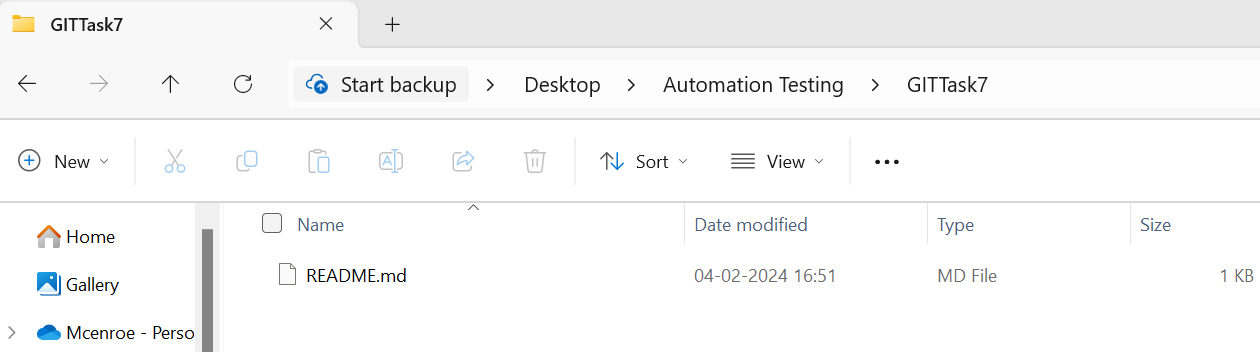
To observe the existence of the .git directory, use command ls -a and use ls -a | grep .git specifically check for the existence of the ‘.git’ directory.

1. Create a README file.

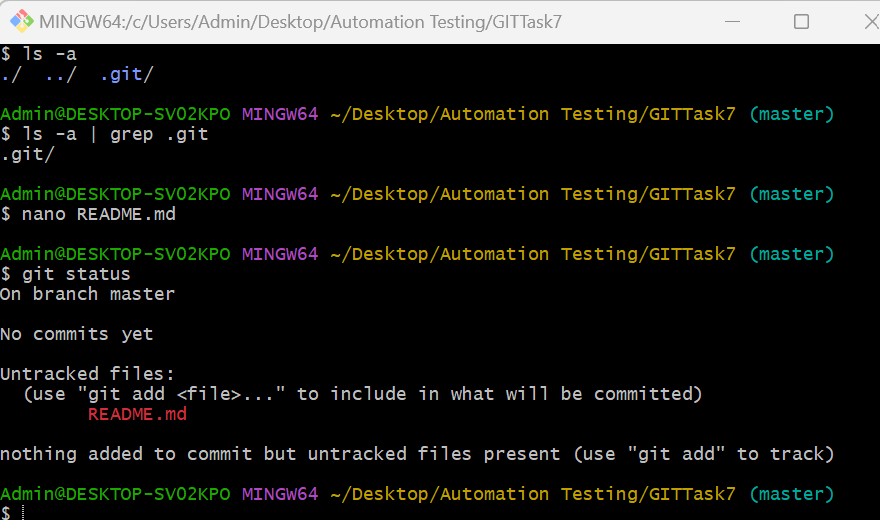


nano README.md command opens a text editor and creates a new file named ‘README.md’.

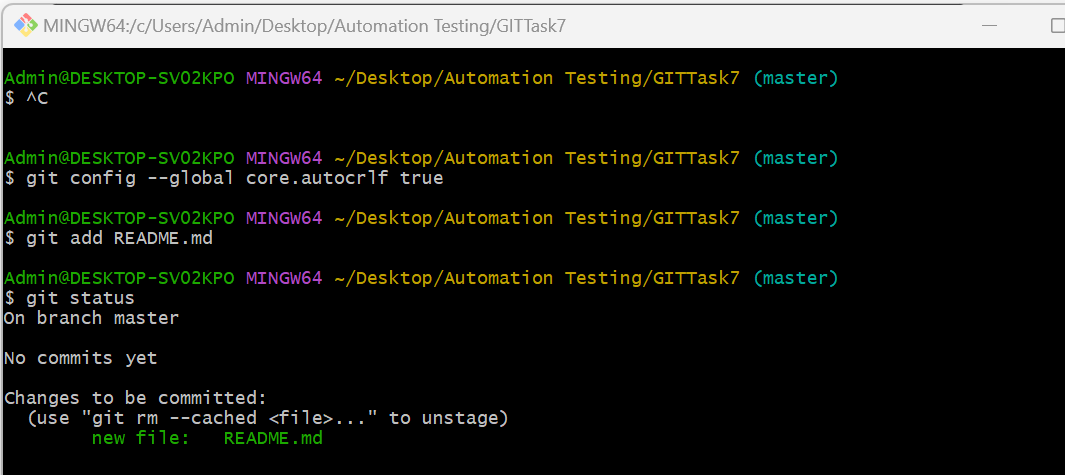
In the text editor, add content to your README file. Save and exit the editor by using key events “ctrl + x”, then press “y” to confirm saving, and press Enter



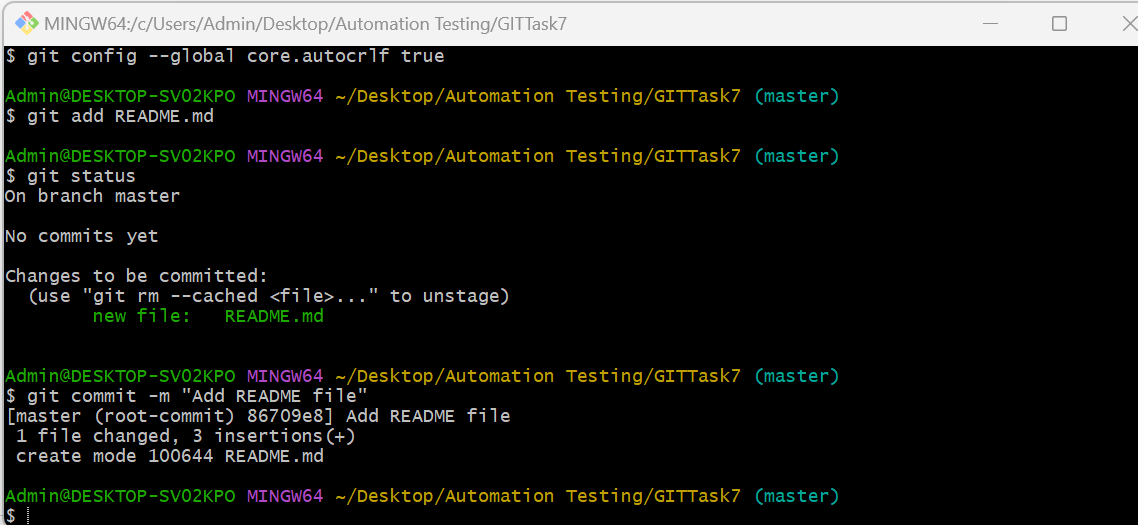
1. Look at the output of the status command. The README created should appear as an untracked file.

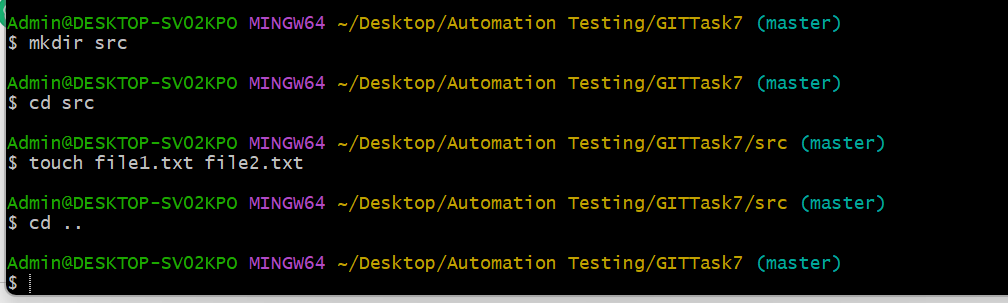


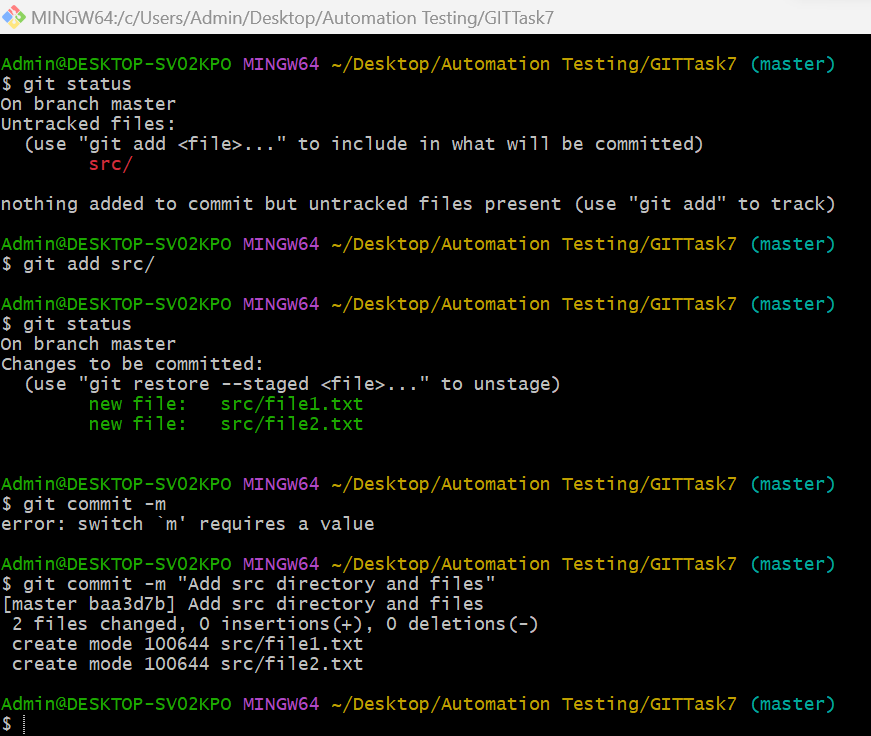
1. Use the add command to add the new file to the staging area. Again, look at the output of the status command.



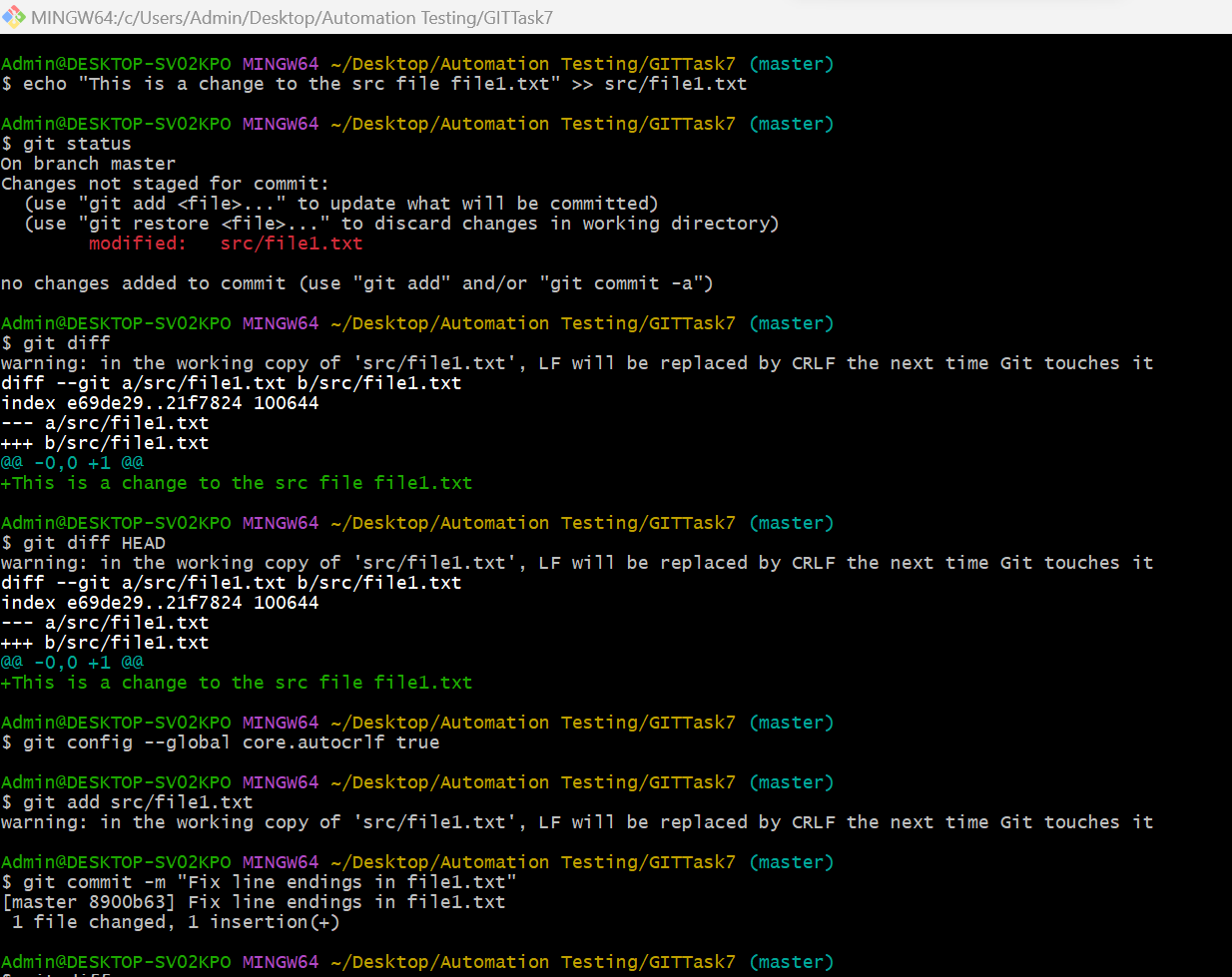
1. Now use the commit command to commit the contents of the staging area.



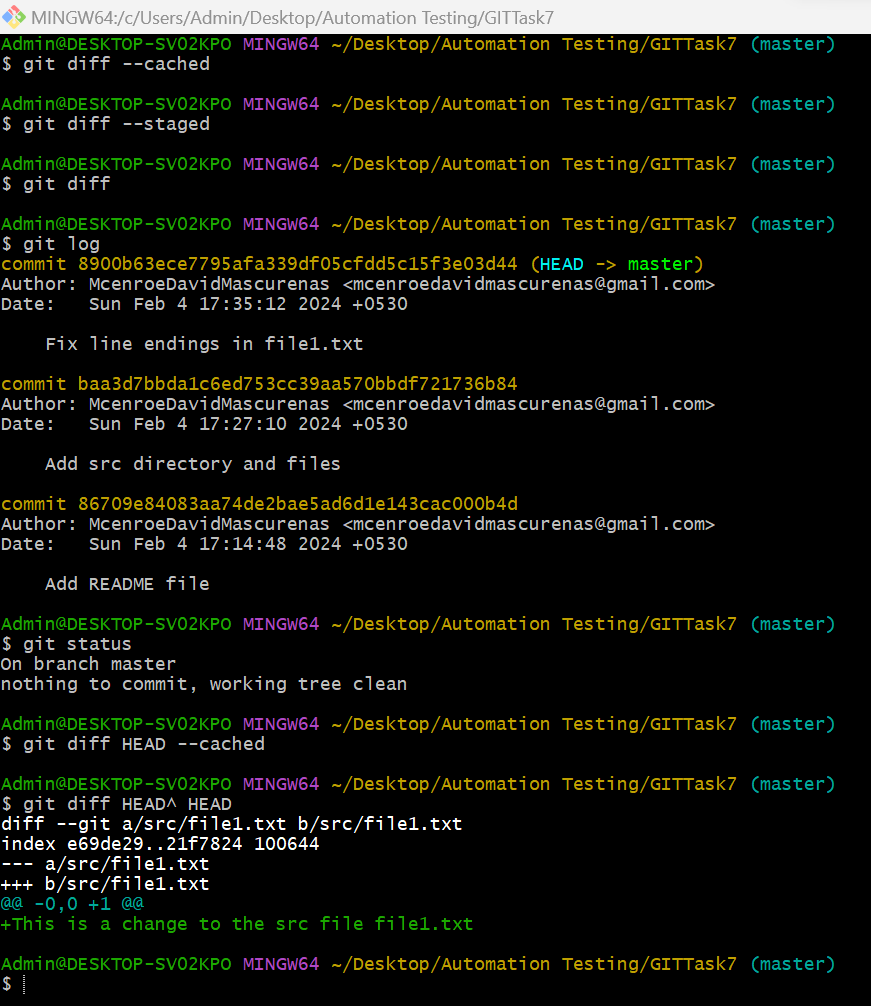
1. create a src directory and add couple of files to it
2. use the add command, but name the directory, not the individual files. use the status command. see how both files have been staged. commit them.



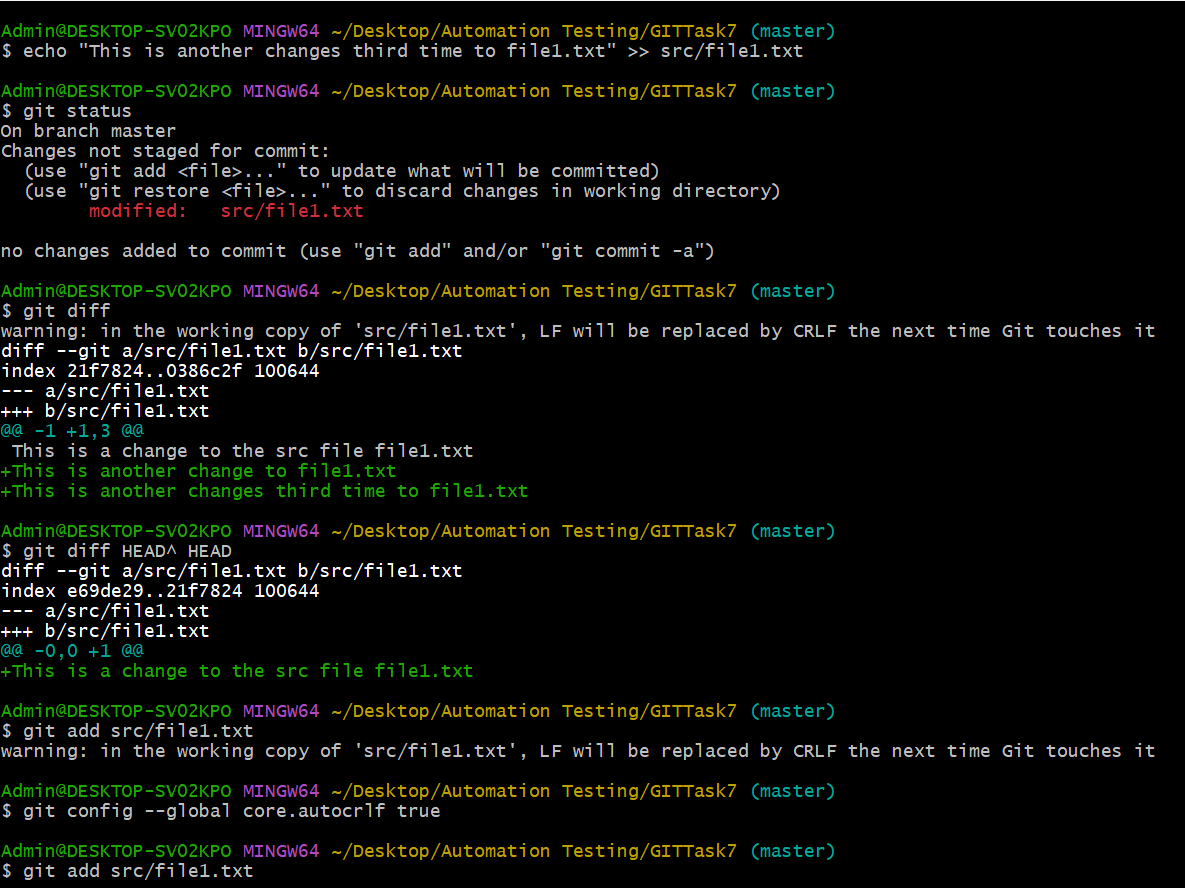
1. make a change to one of the files. use the diff command to view the details of the change.

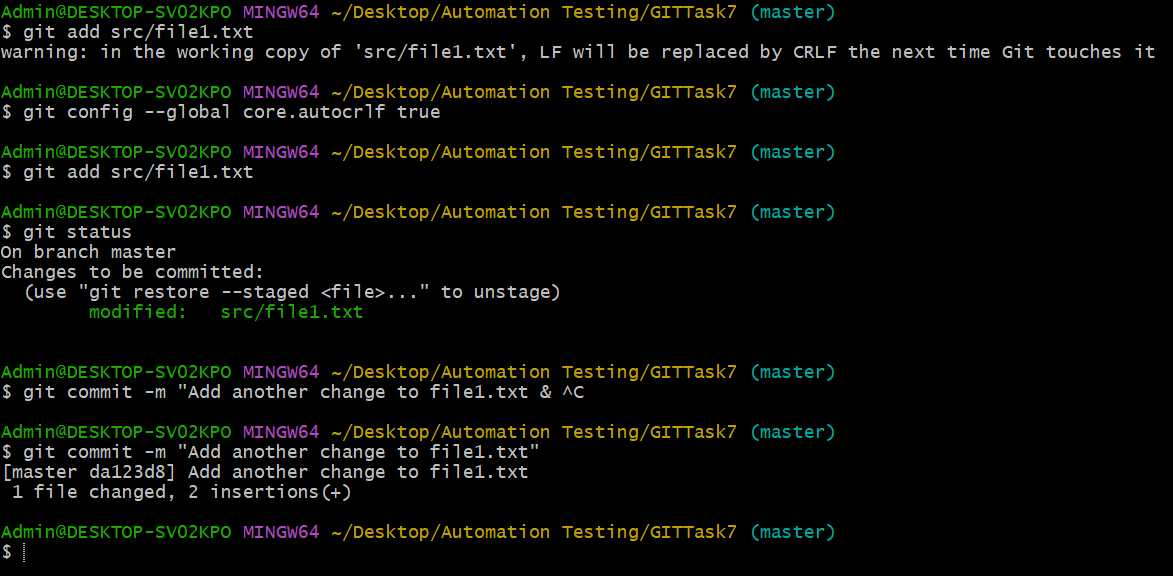


1. Next add the changed file, and notice how it moves to the staging area in the status output. Also observe that the diff command you did before using add now gives no output. Why not? What do you have to do to see a difference in the staging area? (HINT: review the slides if you can't remember)

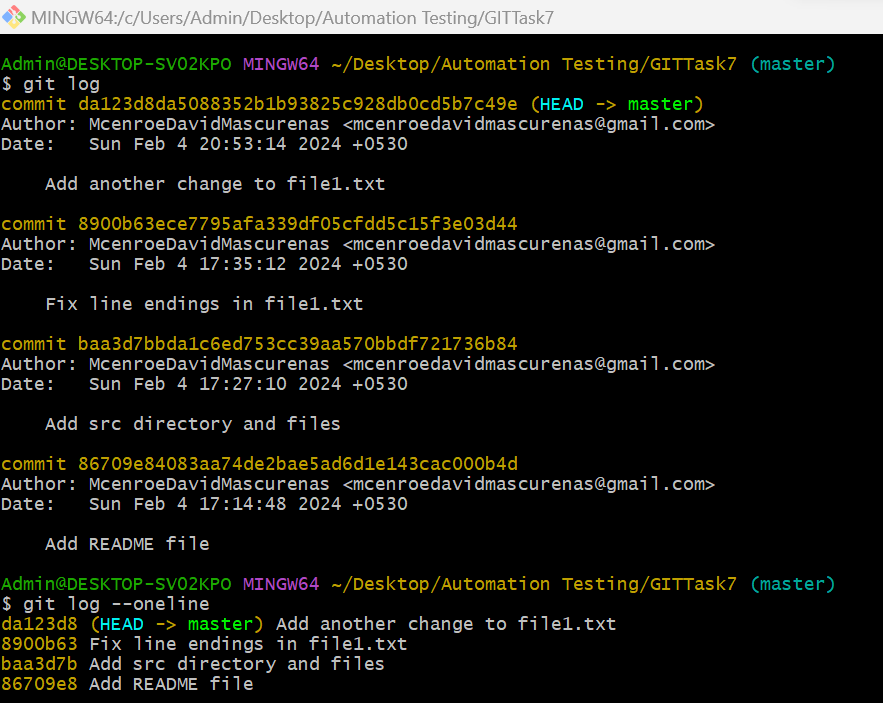


1. Now - without committing - make another change to the same file you changed in step 10. Look at the status output, and the diff output. Notice how you can have both staged and unstaged changes, even when you’re talking about a single file. Observe the difference when you use the add command to stage the latest round of changes. Finally, commit them. You should now have started to get a feel for the staging area.

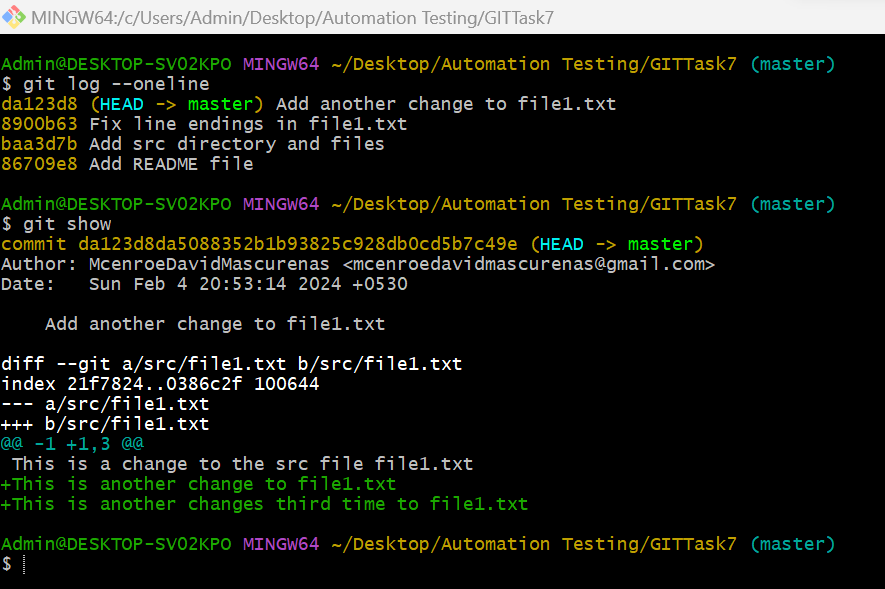




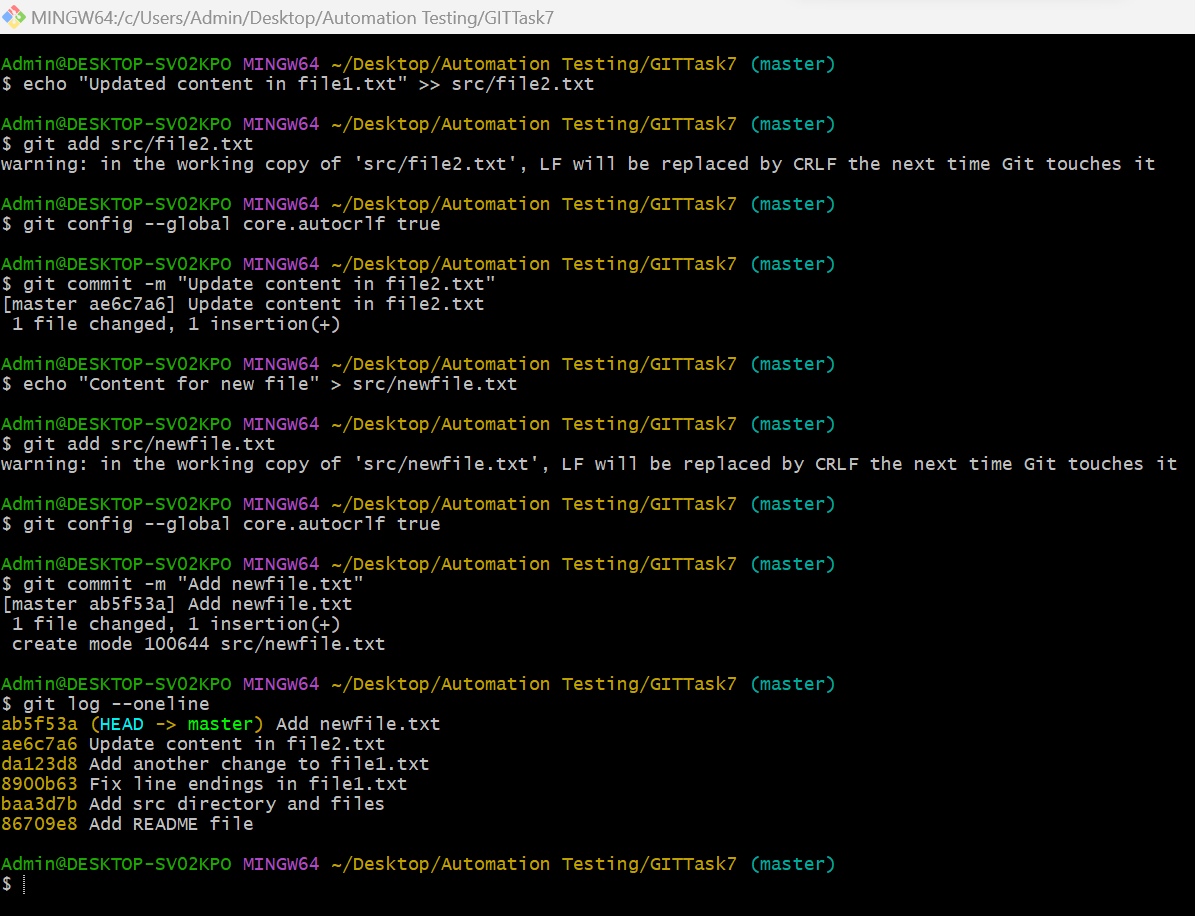
1. Use the log command in order to see all of the commits you made so far.



1. Use the show command to look at an individual commit. How many characters of the commit identifier can you get away with typing at a minimum?



1. Make a couple more commits, at least one of which should add an extra file



Finally clone the local files into the GitHub, by the following steps

