States in GIT:-

**Working Directory**:

This is where you make modifications to your files.

Files in this state are considered to be in the "untracked" or "modified" state.

**Staging Area (Index):**

After making changes in the working directory, you use the git add command to stage these changes.

Staging allows you to group related changes together before committing them.

Files in this state are considered to be in the "staged" state.

**Local Repository:**

After staging your changes, you use the git commit command to save them permanently to the Git repository.

Files in this state are considered to be in the "committed" state.

levels of configuration in GIT:-

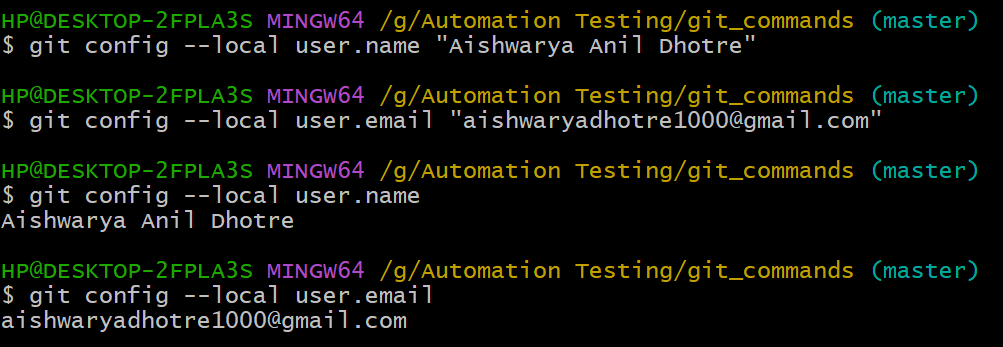
* Repository (Local):-
  + Stored in "/g/Automation Testing/git\_commands/.git/config"
  + These settings are specific to a single Git repository.
  + They are stored in the .git/config file within the repository.
  + Local configurations override global configurations for that specific repository.
  + You can view and edit local-level configurations directly by opening the .git/config file or by using the git config command without any flag, which automatically refers to the local configuration for the current repository.
* User Account (Global):-
  + Stored in - "~/.gitconfig"
  + These settings are specific to a particular user on the system and apply to all Git repositories for that user.
  + They are stored in the user's home directory, usually in the .gitconfig file.
  + You can view and edit global-level configurations using the git config command with the --global flag.

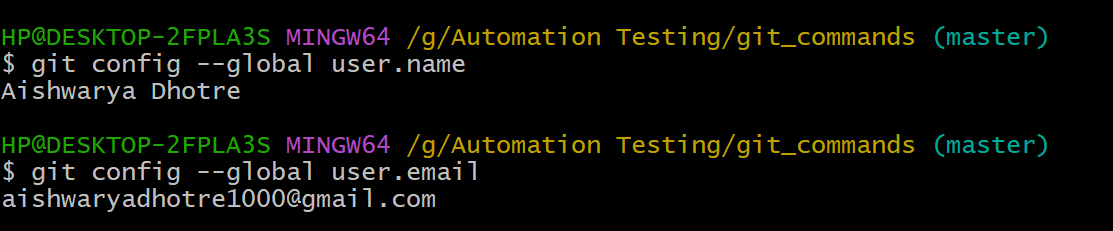
git config --global --edit

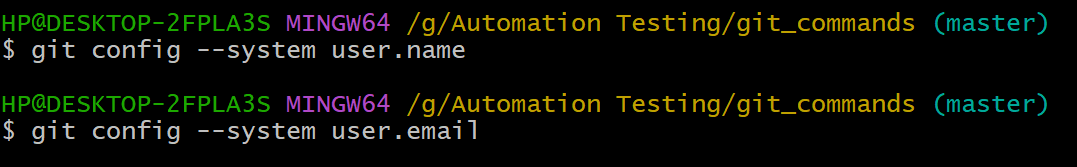
* System Level (Git Installation) :-
  + Stored in - "/etc/gitconfig"
  + These settings apply to the entire Git system and are shared among all users on the machine.
  + They are stored in the gitconfig file within the Git installation directory.
  + You can view and edit system-level configurations using the git config command with the --system flag.

git config --system --edit

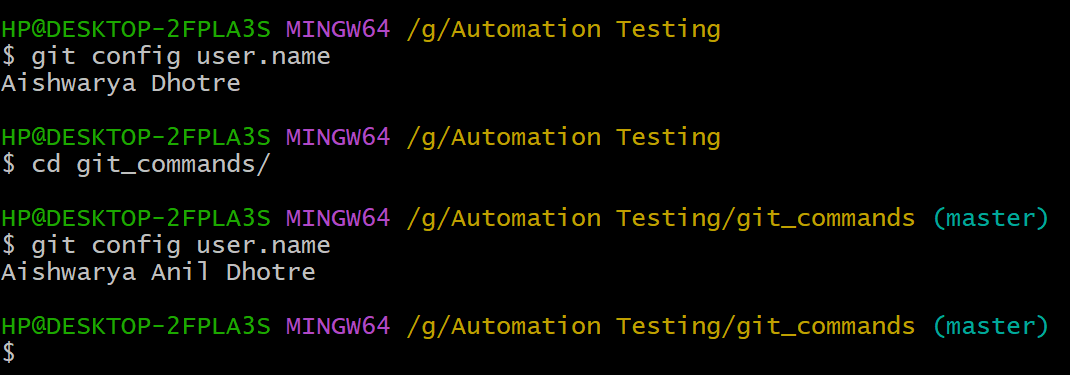
**Config user name and password**





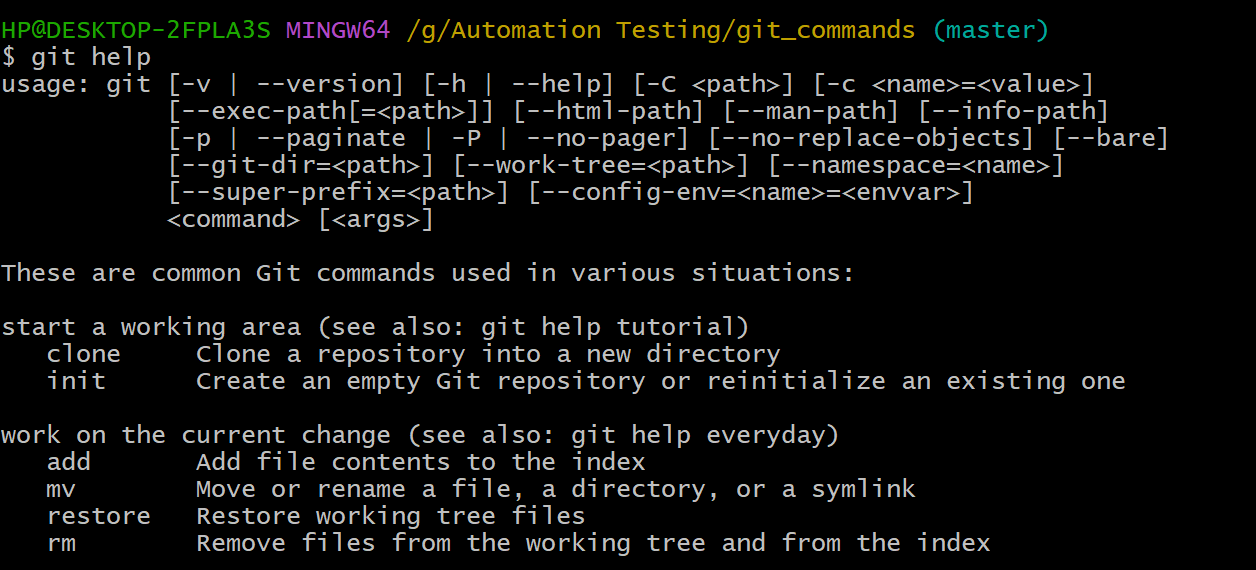


* The priority of configuration levels is such that local configurations take precedence over global configurations, which, in turn, take precedence over system configurations

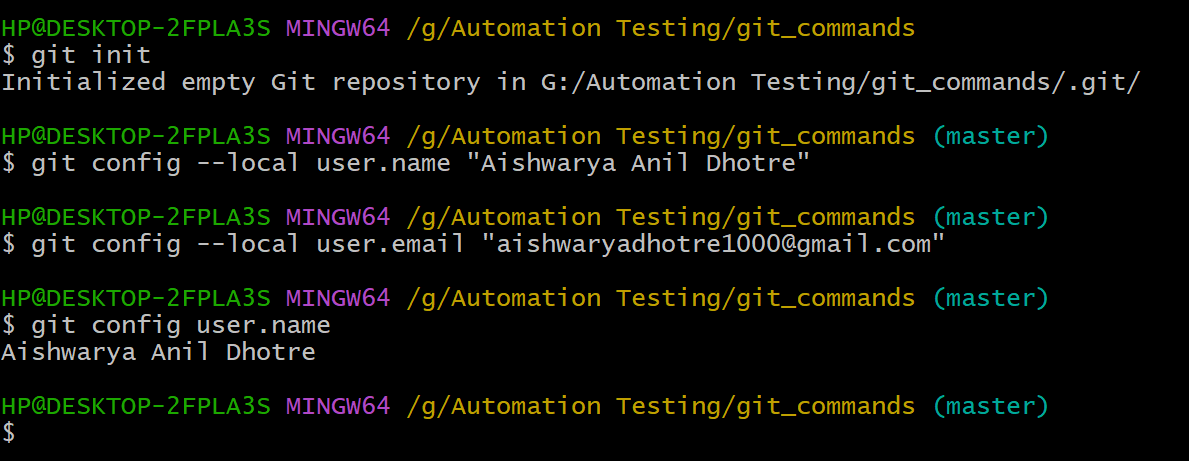


**Git Commands**

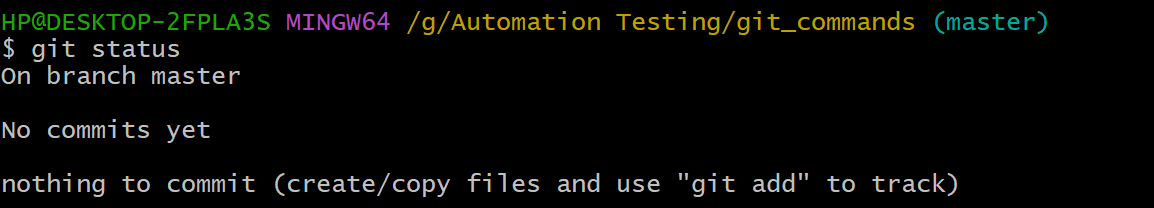
* Git help



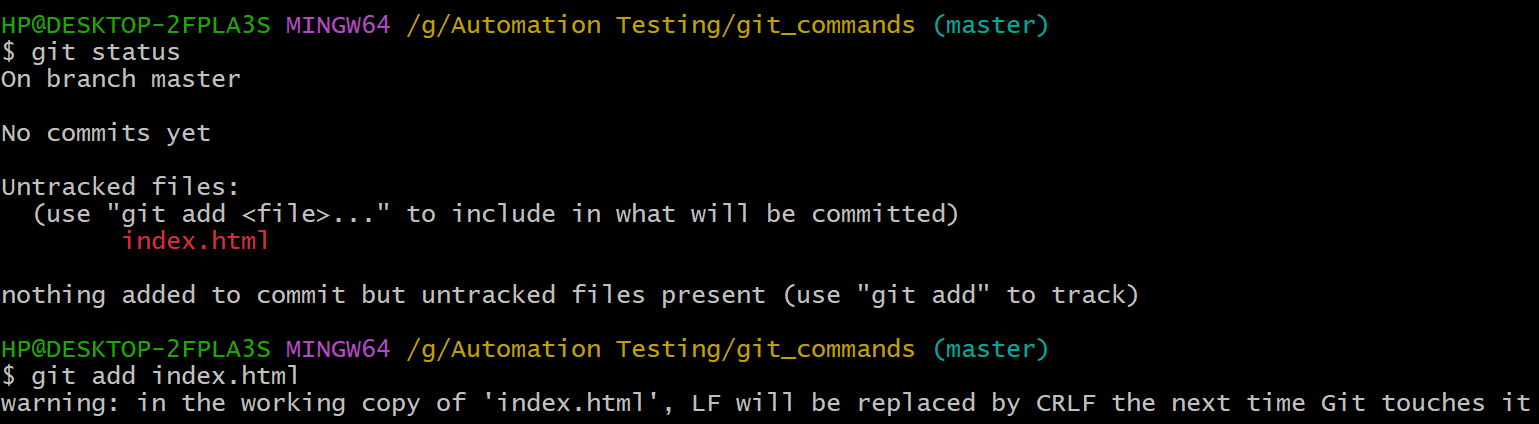
* Git init

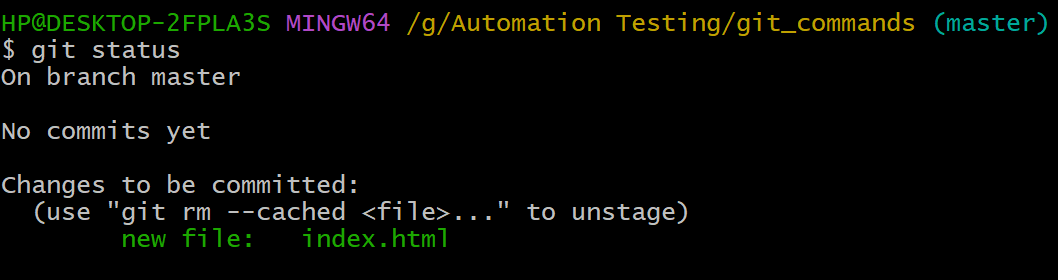


* Git status

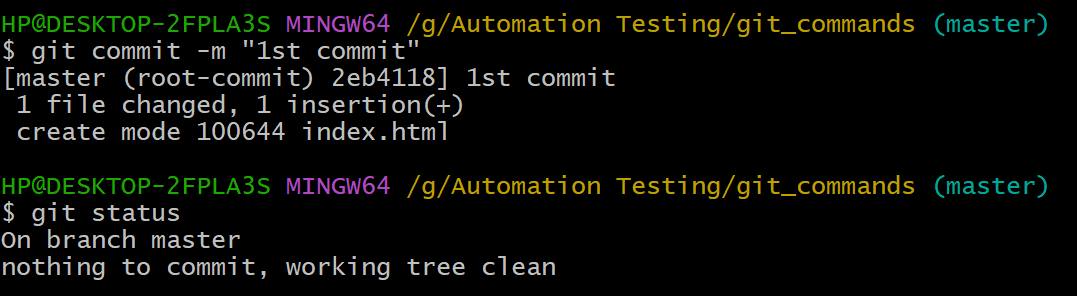


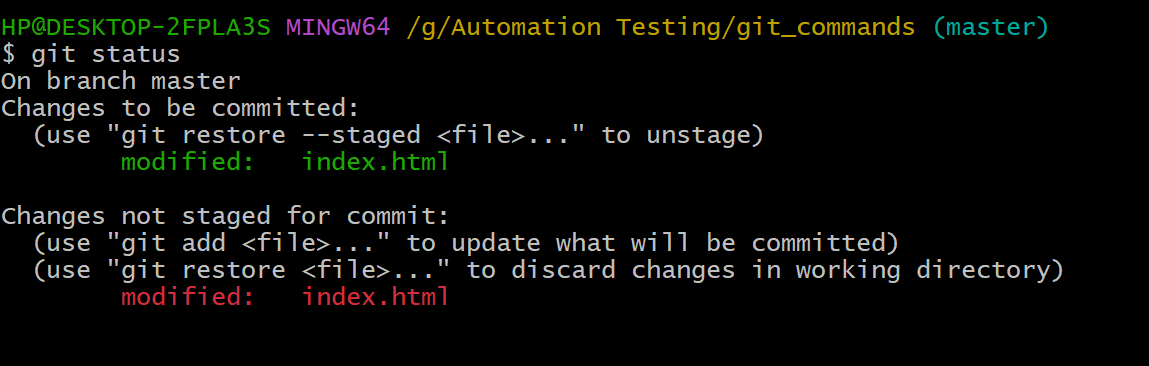
* Git add



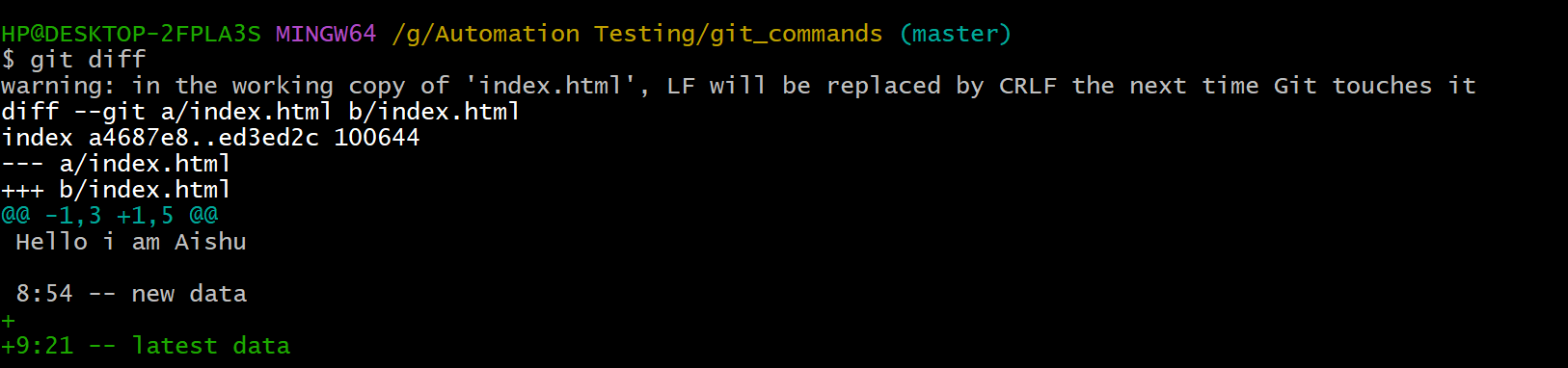


* Git commit

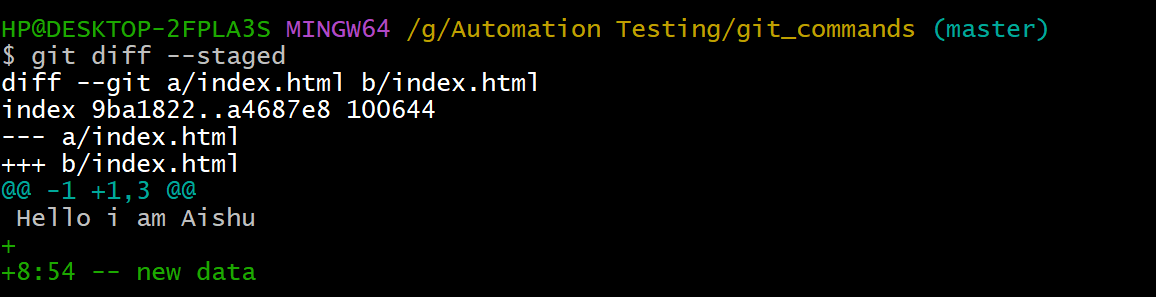




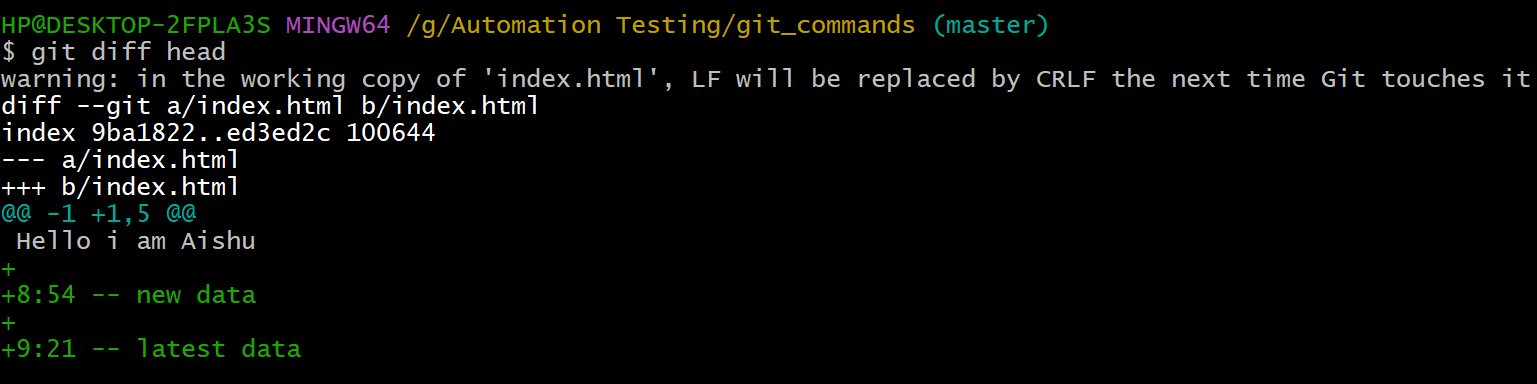
* Git diff: - compare the changes in the working area & staged area



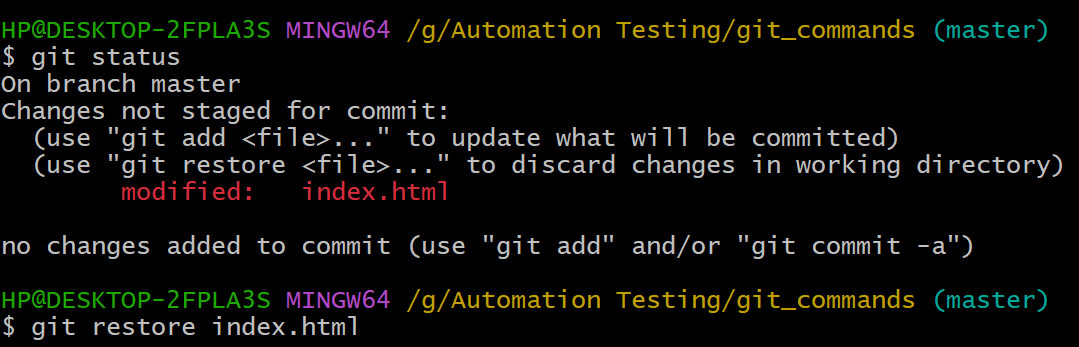
* Git diff –staged: - compare the changes in the staged area and repository area



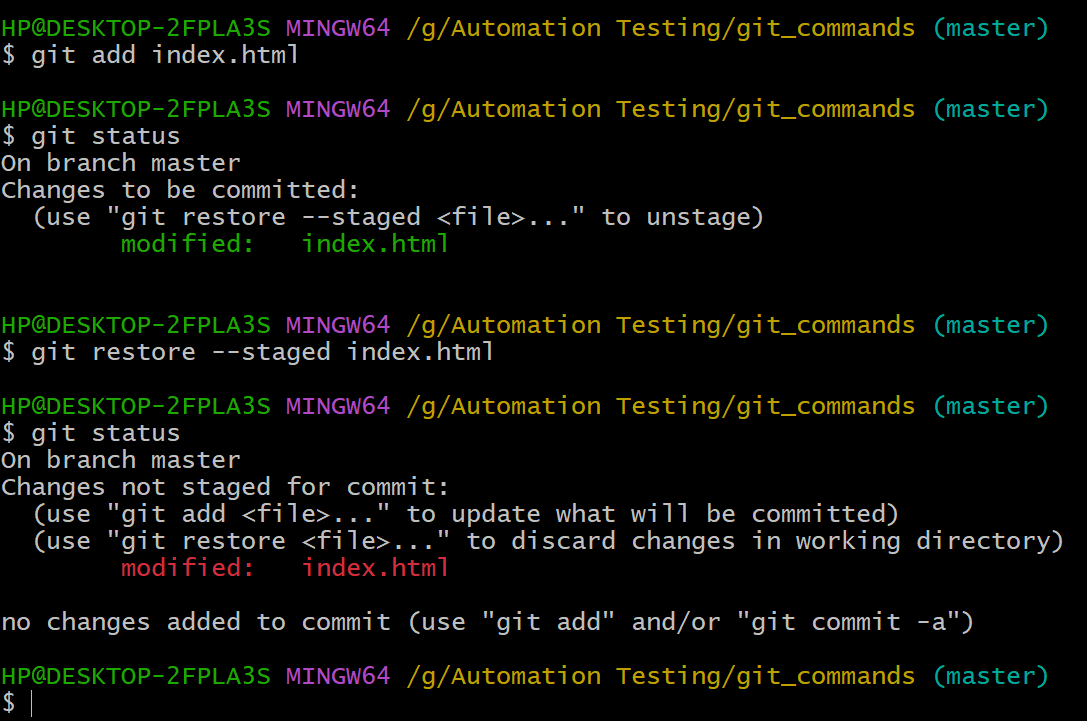
* Git diff head:- compare the changes in the working area and repository area



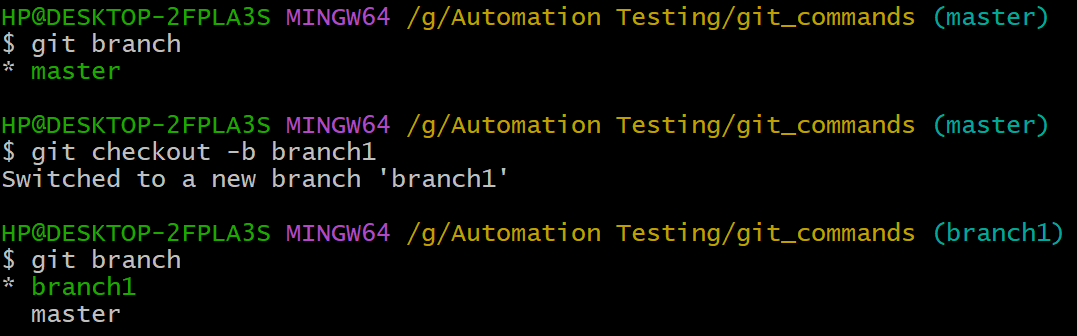
* Git restore



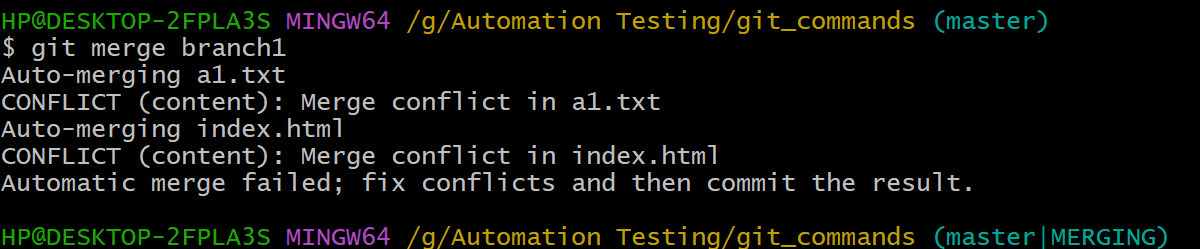
* Git restore –staged



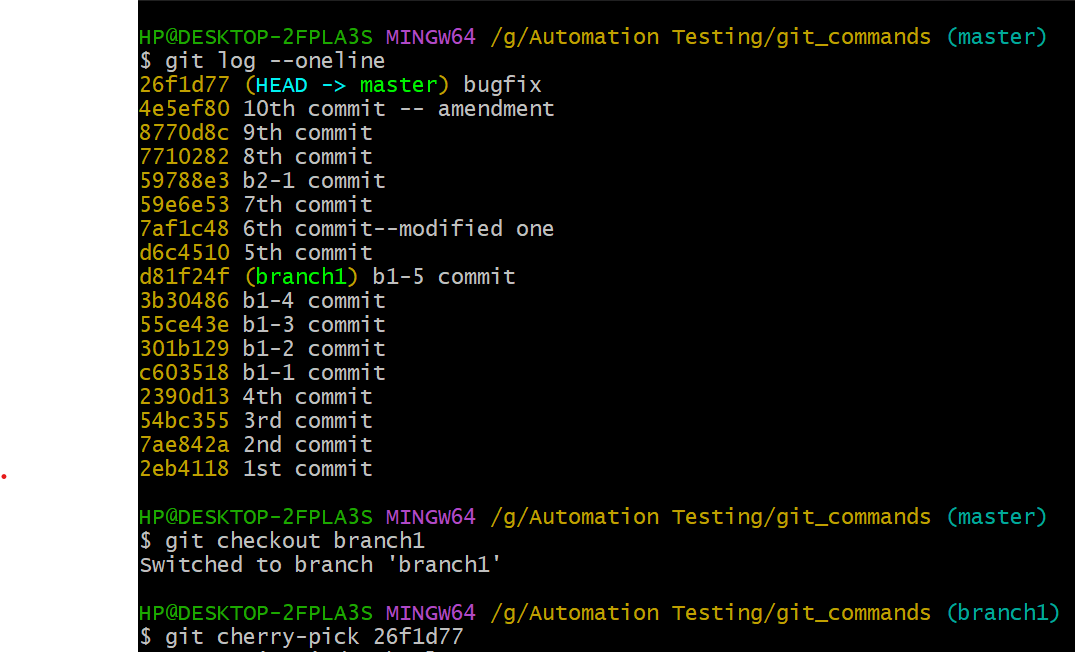
* Git branching
  + Git checkout -b branch1
  + Git branch



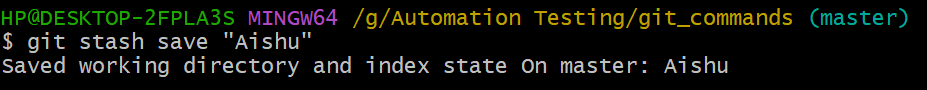
* Git rebase branch-name - rebase doesn’t add the extra commit.
* Git merge branch-name - merge the desired branch into the current branch & add one extra merge commit

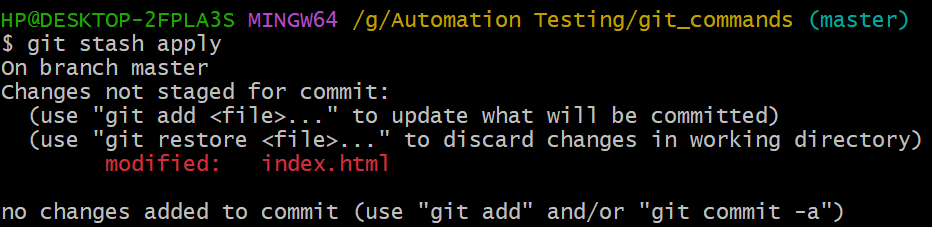


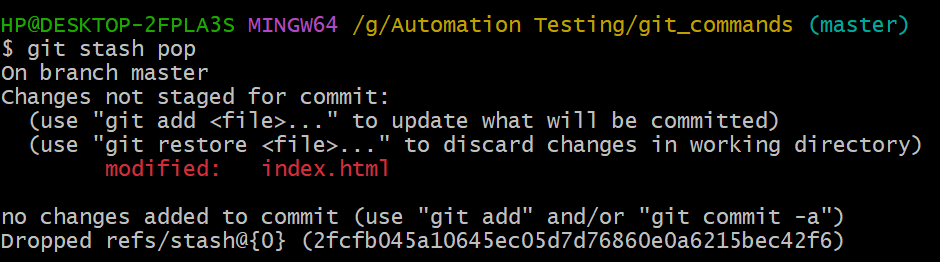
* Git commit --amend : – chnage the latest commit
* Git cherry-pick hash – apply specific commit from one ranch o another branch

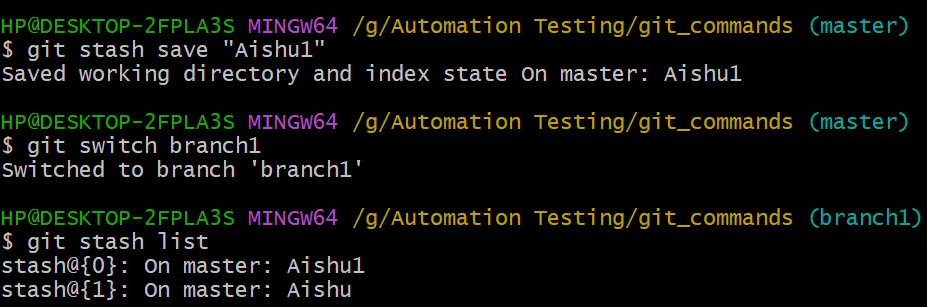


* Git stash save “stash-name”– save the changes in the working directory to stash
* Git stash pop – remove & apply the stash from the stash list
* Git stash apply - apply the stash without removing it from the stash list
* Git stash list – list the contents in the stash
* git stash apply stash@{1} – apply the specific stash



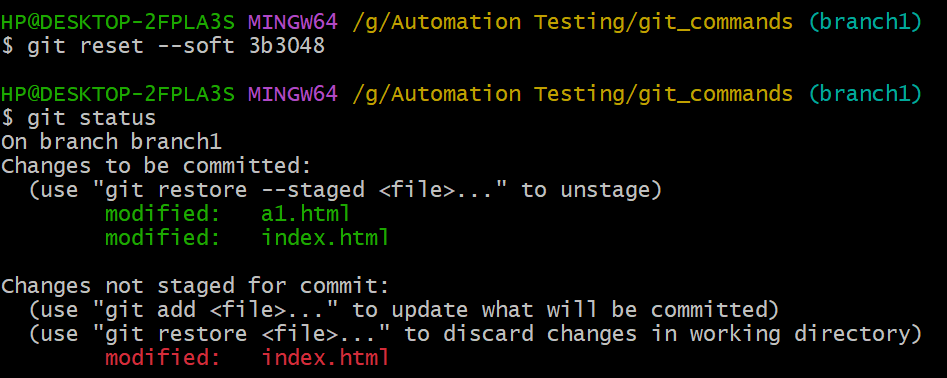


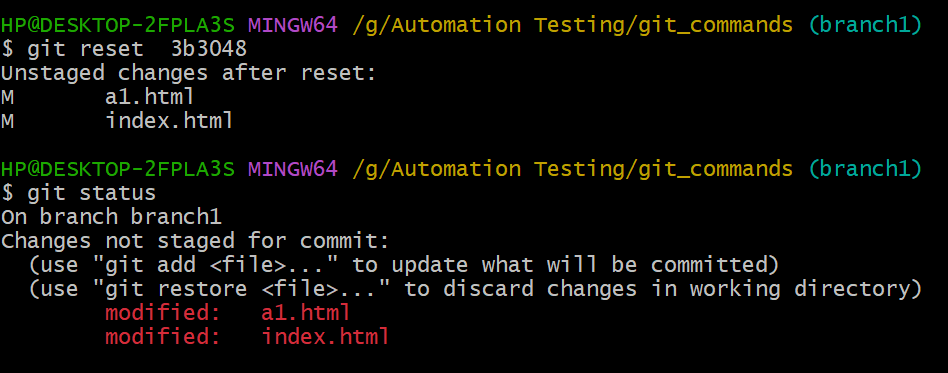


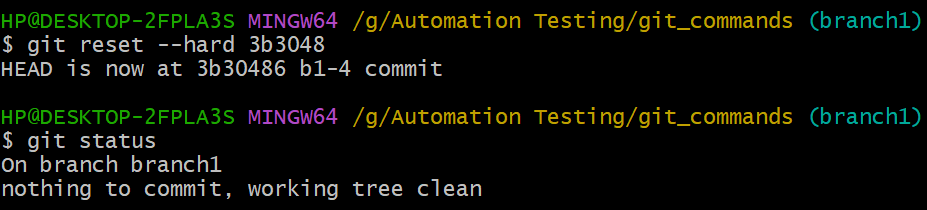




* Git reset :- default flag is mixed
* Soft : Moves the branch pointer to the specified commit.Leaves the changes staged (changes will not be lost). keeps the changes in the working directory as it is .
* Mixed : Moves the branch pointer to the specified commit.Unstages the changes (changes are not lost). keeps the changes in the working directory as it is .
* Hard : Moves the branch pointer to the specified commit.Discards all changes (working directory and staging area).





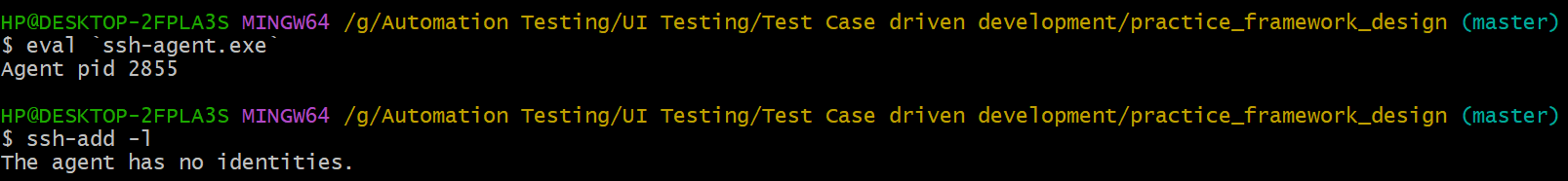


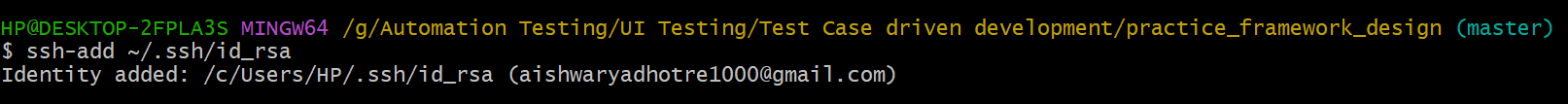
* Git reflog - This command will show a list of recent actions and their corresponding commit hashes and references. It's a useful tool for tracking changes and can be particularly handy for recovering lost commits or branches.

**Push changes to remote repository:-**

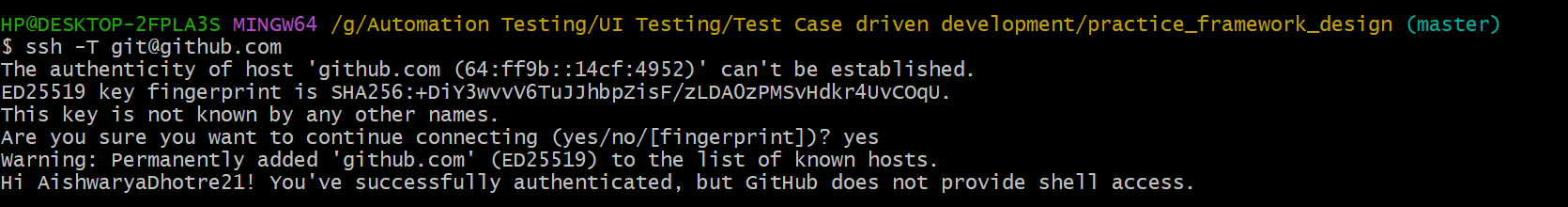
1. Create github repo
2. Generate ssh keys
3. Add public key to github
4. Initialize the git in the working repository

* Start ssh-agent.exe service
* Add private key to ssh-agent

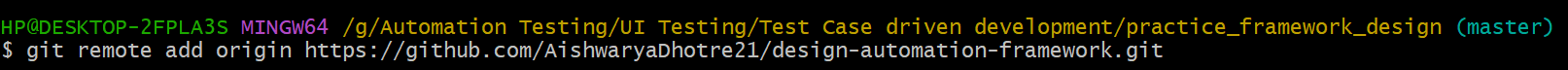


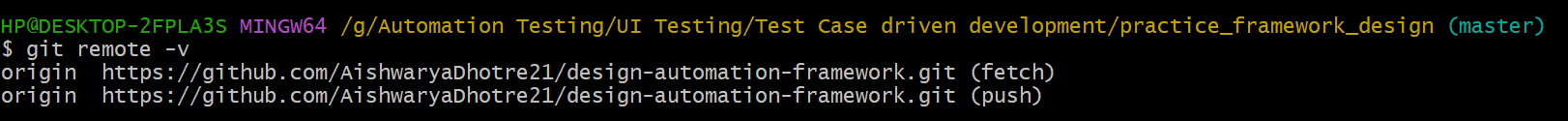


* Authenticate with github

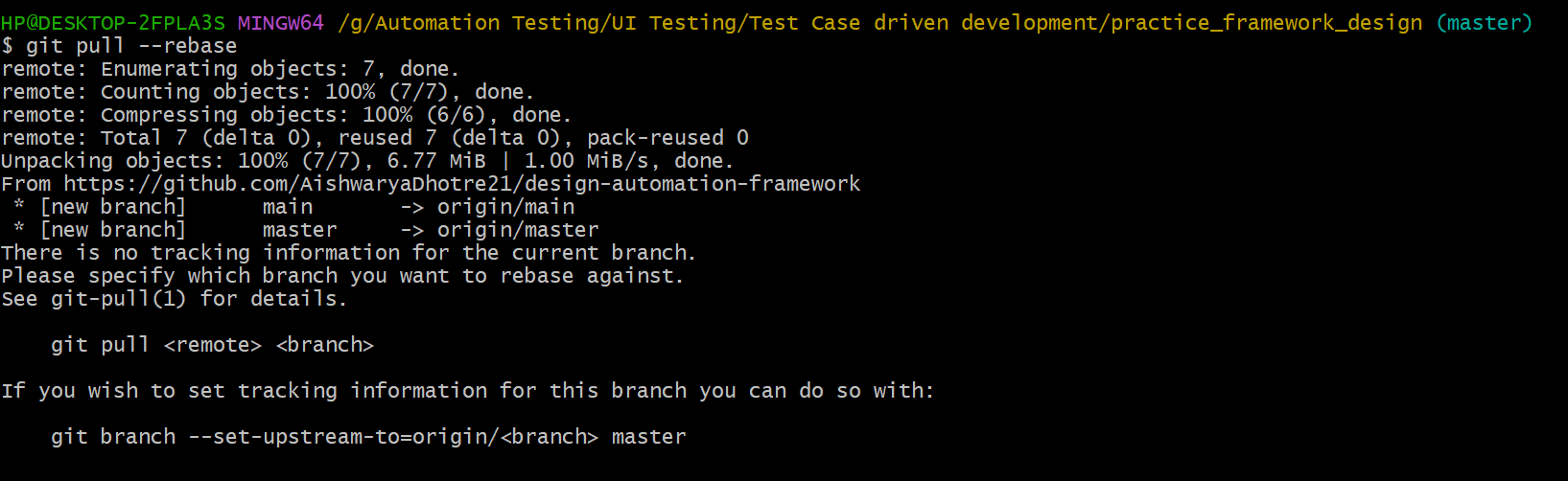


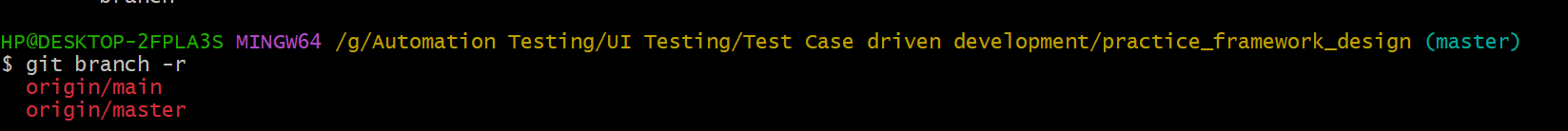
Add remote url

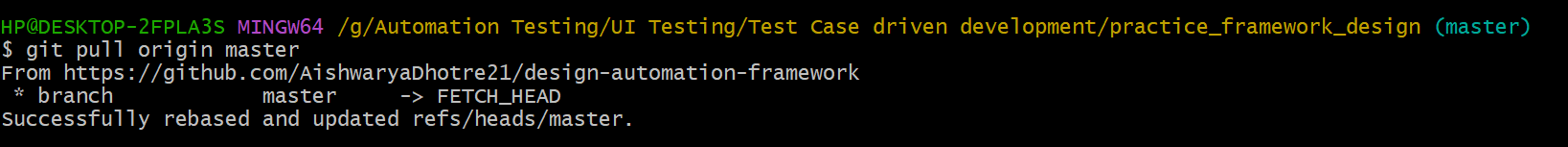




Rebase the remote branch on local branch







Push the code

