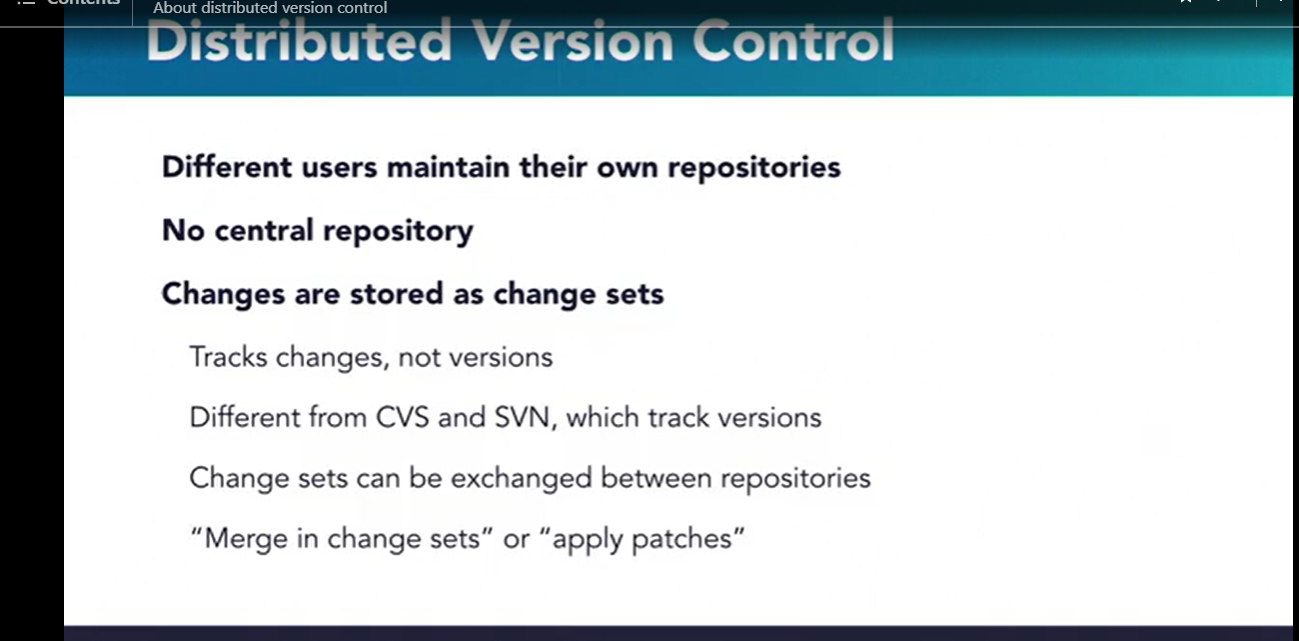
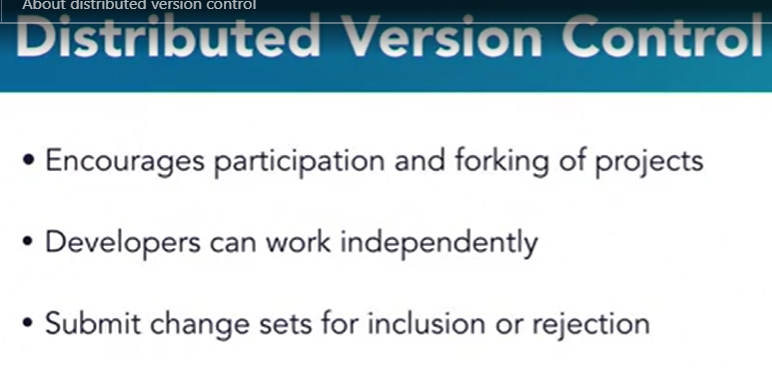
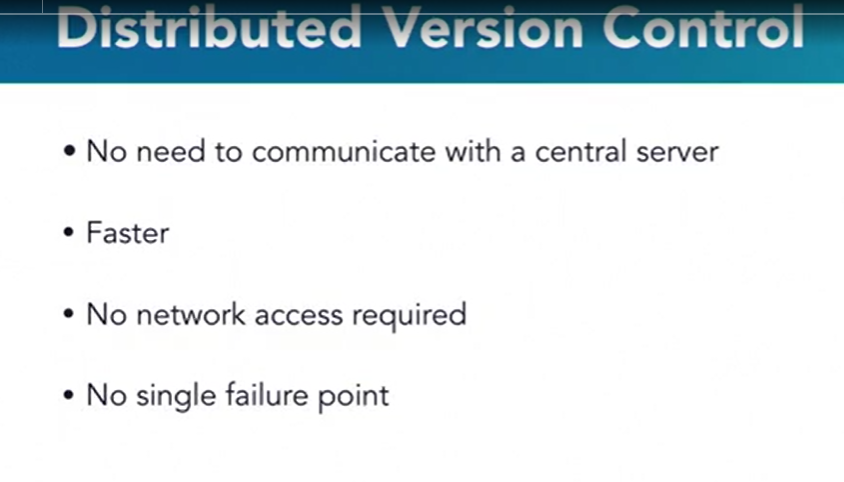
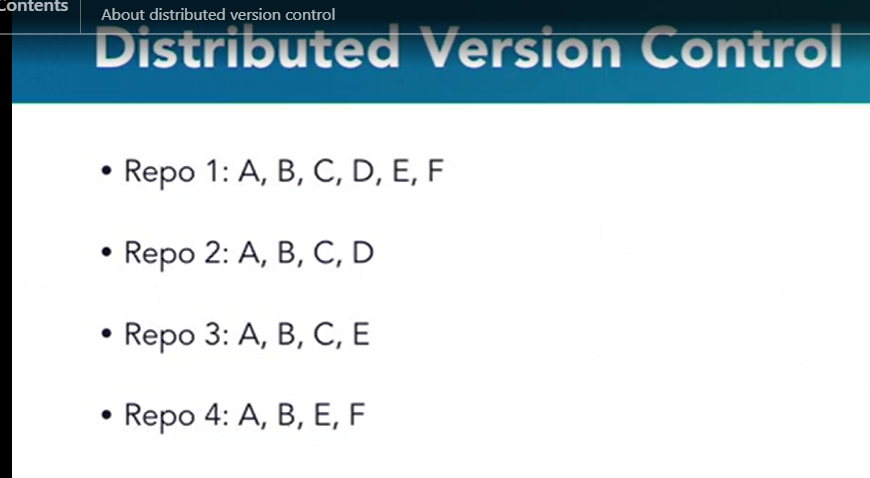
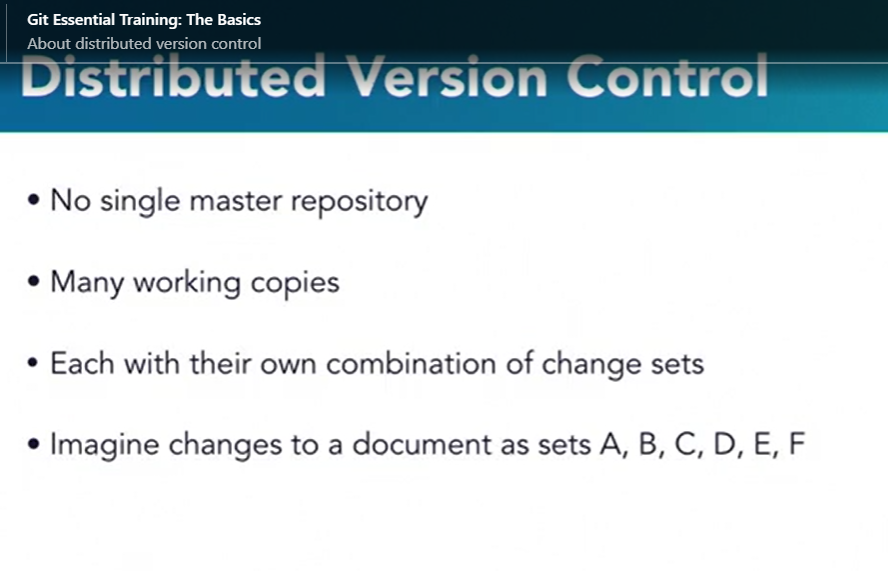
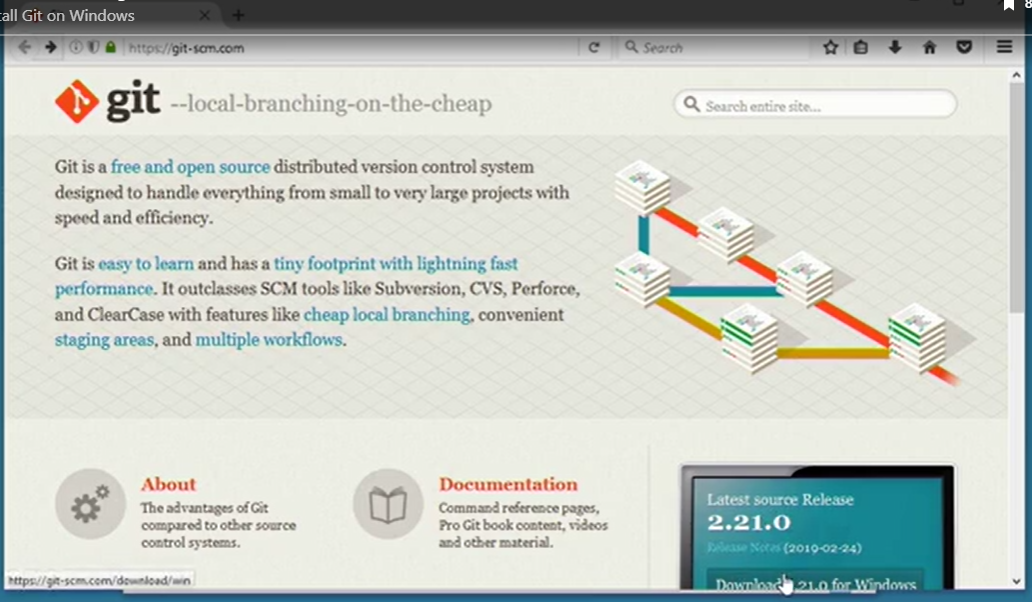
* Git is a Distributed version control system(VCS) or Source Code Management(SCM) Tool which keeps track of changes especially text changes n which can be used to manage code.

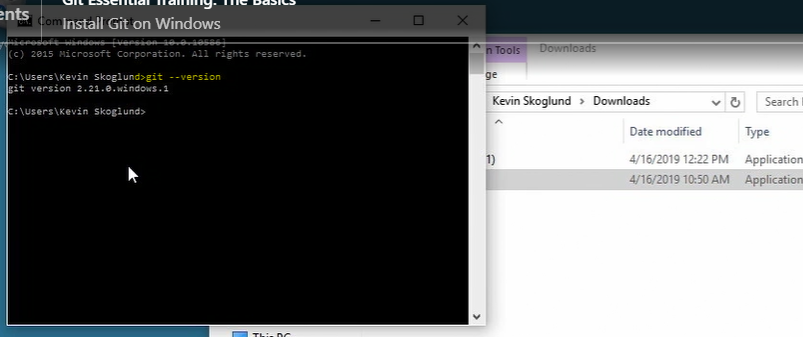








Install the git with default provided selections,once complete check the git installation via command prompt by typing git –version in command prompt Or in git bash by typing git –version or git version.



Note:To increase the font size of word in git bash press ctrl and +

Similarly to reduce the font size of word in git bash press ctrl and –



Git configuration can be applied at different levels:

1)System:these configurations can be applied to all system users.

Command: git config --system

2)User or global:these configurations can be applied to specific user.

Command: git config --global

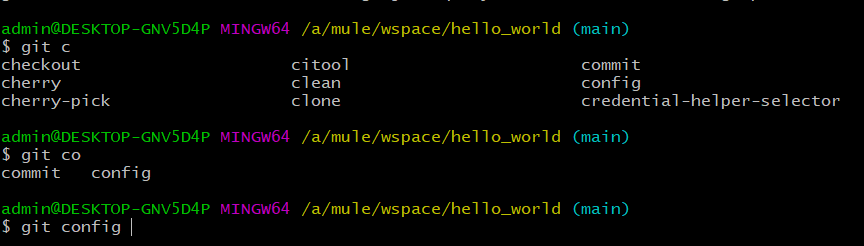
3)Project:these configurations can be applied to specific projects .git folder.

Command:git config



Git allows its commands to be auto populated if we need to do that.Windows supports auto completion by default but mac n linux needs to perform some extra activivties to allow git to auto complete the command.We use tab keyboard button in all os to auto complete the command.

Let say we want to write git config command but we forgot the statement ( like whether it is config or configs or configuration or configurations ) then what we can do is we can write git c n then press tab button two times(to see the available command options) so it will show available commands starting from c.if you type git co n then press tab button twice then it will show available commands starting from co i.e commit n config.if you type git con n then press tab key once(since only one command will be there with con). it will auto complete the git config



Git Three Stage Architecturer:

Git used to track the changes,so changes has to be gone thru from 3 different stages:

1)Present Working Directory:The changes are initiated from this stage where changes are actually start.

2)Staging Area:once we done the changes in present working directory they need to be forwarded to staging area this we can do by adding the changes from present working directory to staging area.use this command

git add .

. adds each changes in staging area.to add particular change let say we done the changes in first.text in pwd if we want we can add only changes done to first.txt to staging area by writing below command

Git add first.txt

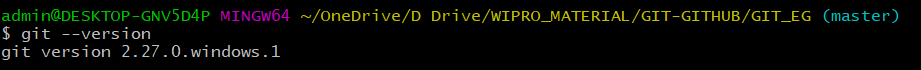
3)Git Repositories:once we add the changes to staging area ,final stage is commiting it to repository.that we can do by following command

Git commit -m “commit\_msg”

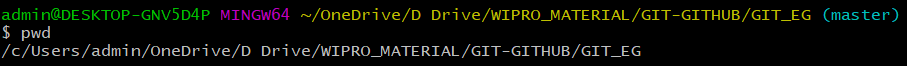
Where -m stands for message

Git Commands to run on git bash:

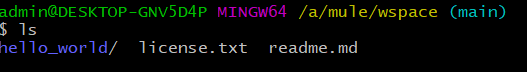
* git --version:-tells you the git version which is installed on your system.



* pwd:gives present working directory name.

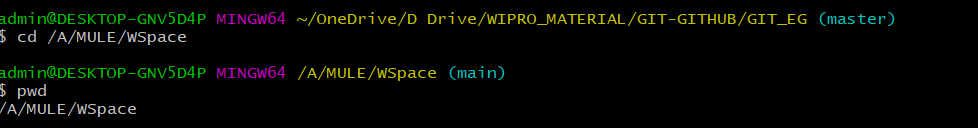


* ls:shows all files n folders from present working directory.



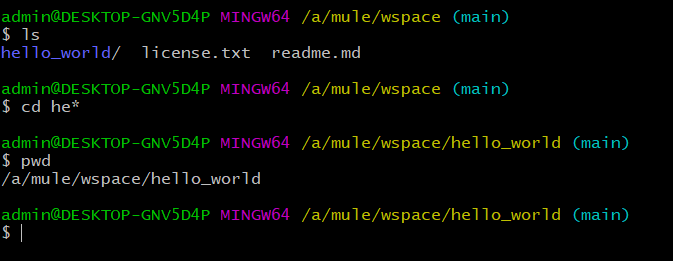
In blue it shows folders n in white shows files.

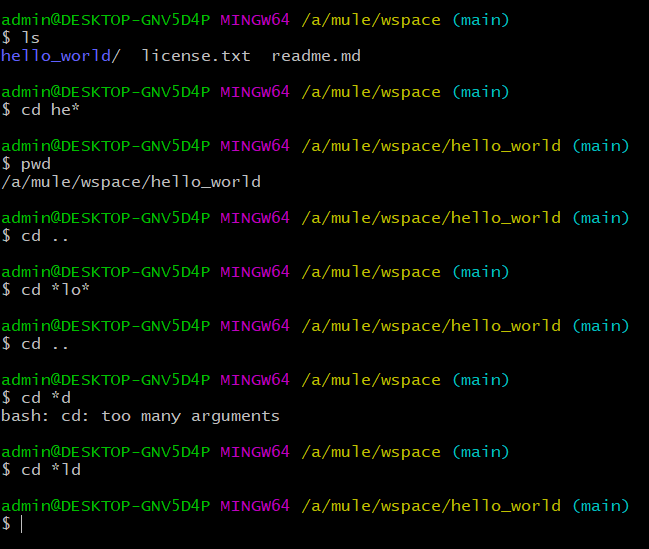
* cd /<<directory\_path>>:changes the directory(folders).used for jumping between directories(folders).



we can use wildcard character like \* also like cd he\*,cd \*lo\*, cd \*ld

Make sure only exact matched folder name should be entered then only it will be able to switch between the folders, other wise will show error like too many arguments.

\\

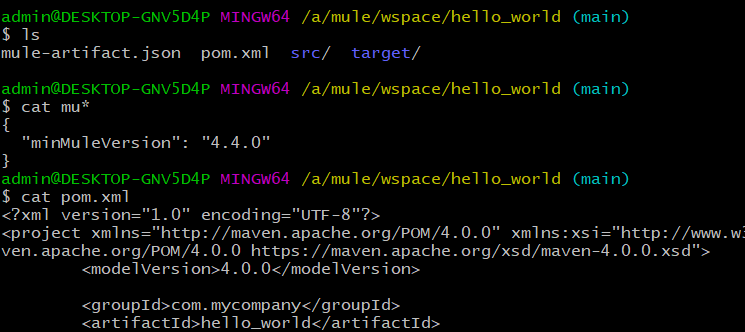


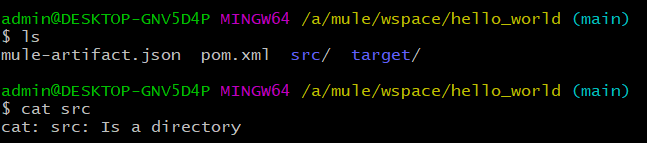
If you want to go reverse in folder level then you can use cd .. command.it will take you to the backed folder.let say currently we are using /a/mule/wspace folder n if I want to traverse back let say to /a/mule then I can use cd ..

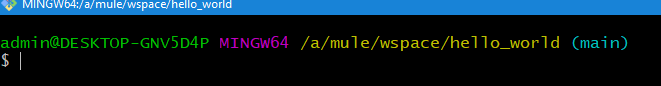
Note:Cd command is only used for jumping from one folder to another.If we write cd first.txt then it gives error like first.txt is not a directory.

* Cat <<filename.ext>>:shows content/body of the file n not the folders.we can use wildcard character like \* also like cat li\*.txt,cat \*lib\*,cat \*xt.

Make sure only exact matched file should be one then only it will show the body of the file other wise will show error like too many arguments.



If we say cat folder\_name then it will give error like folder\_name is a directory. 

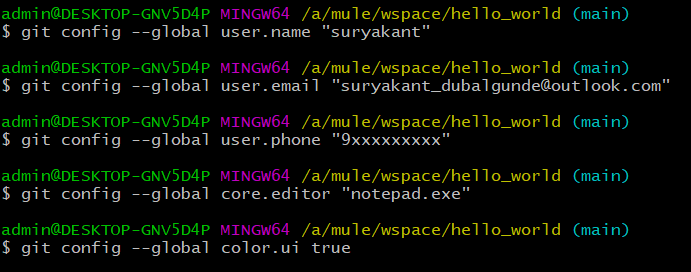
* Clear: clear the screen on git bash .it will open a fresh window 
* git config or git config --system or git config –global:It is the command that we will use to set view n edit/delete/removing configuration the parameters.

For setting the values at different levels: use

git config –system user.name “suryakant”

git config –global user.name “suryakant”

git config user.name “suryakant”



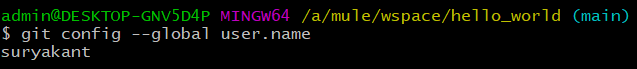
For viewing the values at different levels: use

Particular value: use

git config –system user.name

git config –global user.name

git config user.name

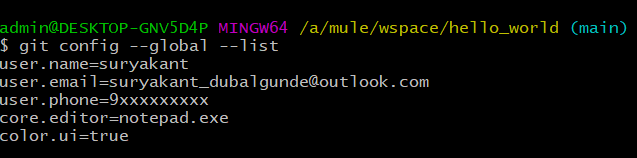


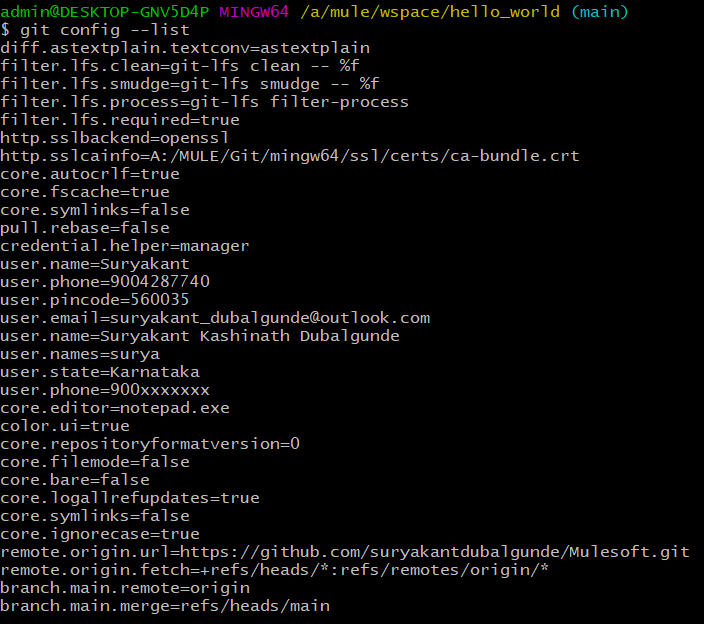
all values at different level:use

git config --system --list

git config --global --list

git config –list(shows all levels config values)



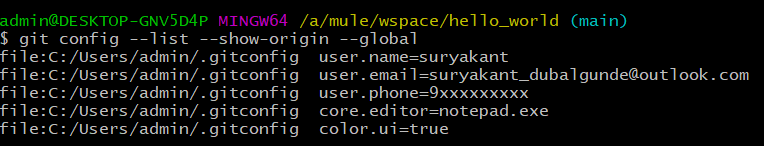


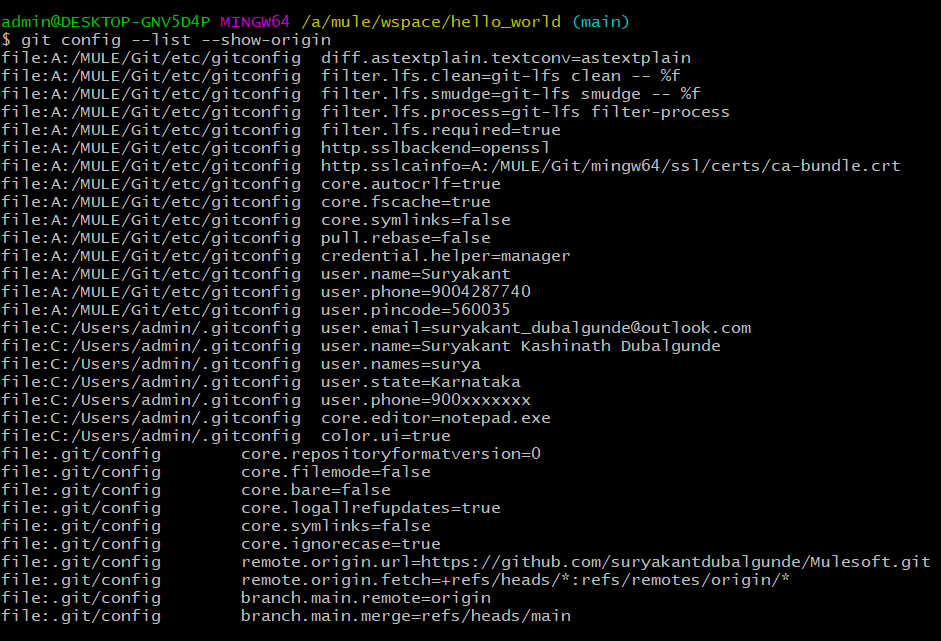
For viewing the values at different levels with label name to identify which level configuration: use

Git config --list --show-origin --system

Git config --list --show-origin --global

Git config --list --show-origin(shows all levels config values)





system config= which contains Git/etc/gitconfig in a label

user or global config= which contains user home directory like c:/Users/admin/.gitconfig in a label

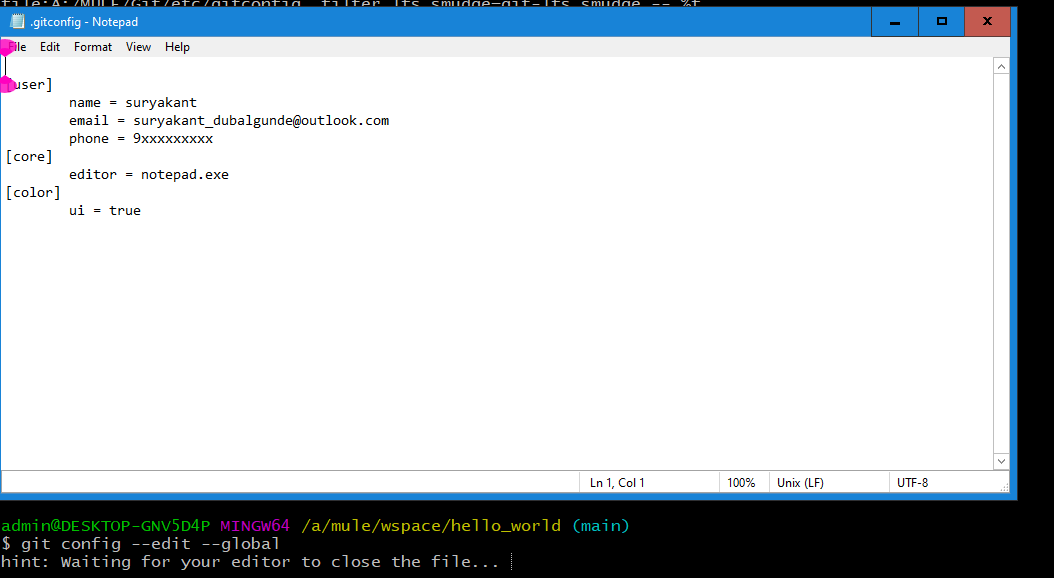
project config= which contains .git/config in a label

For editing/deleting/removing the values at different levels: use

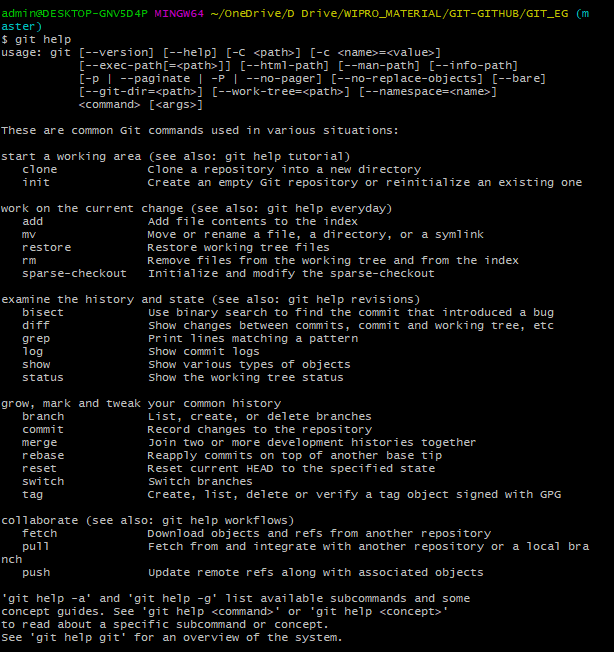
git config –system --edit

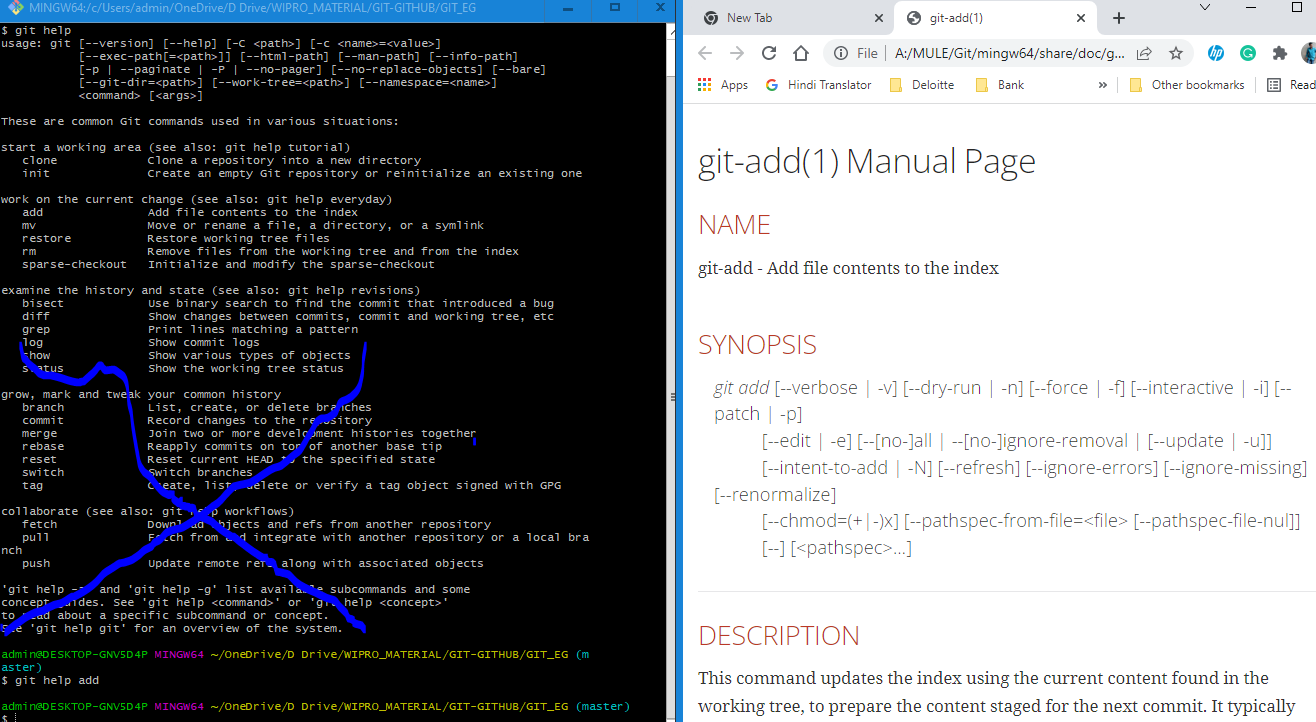
git config –global --edit

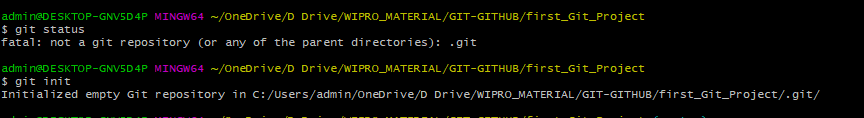
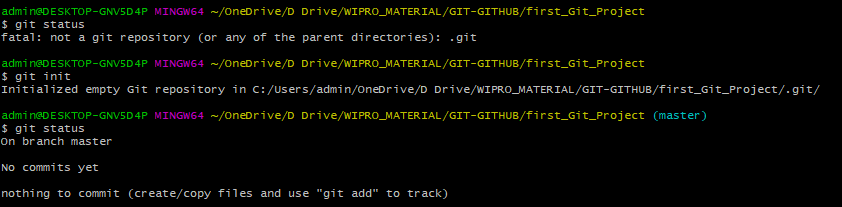
git config --edit



--edit command opens up notepad where we can add new entries,delete the entries or do some changes in the entries.Why notepad opened is because we have set core.editor=notepad.exe as global config that’s why.

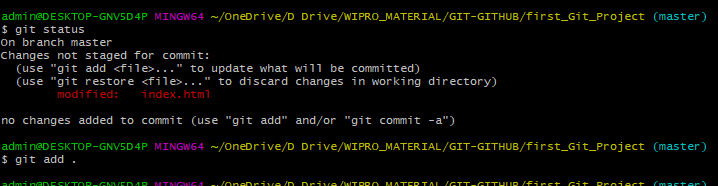
* Git help <<command\_name>>: this command will help you to understand how to work with the git.only git help command will show all available commands like add,clone etc. 

git help <<command\_name>> will help you to understand about a particular command.let say we wrote git help add then it will provide information abt add command in git. 

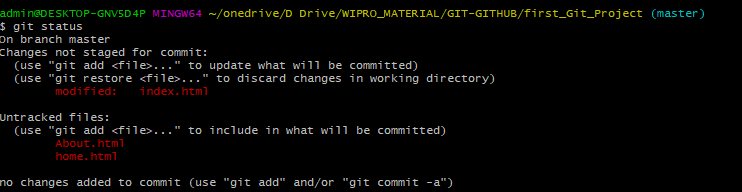
* git init:Used for intializing a git in a project folder.This command creates .git folder in our present working directory that is used to track the changes.if we delete this .git folder or by mistake we throw git init command again it will reinitialize the .git folder again and previously tracked changes will not be available.
* git status:command is used for checking the status of present working directory like if there any files which are not tracked ,any new file came? Like that tracking. 
* git add . :command is used for adding present working directory changes to staging area. We can use

git add . -------------to add all changes made in present working directory to staging area.

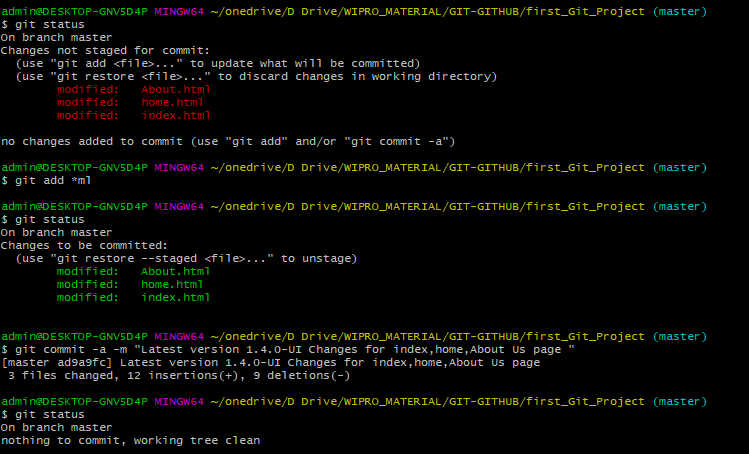
git add <<filename.ext>>-----------------to add only filename.ext change from present working directory to staging area.



Note:If we are adding something new let say we add two new files in our project so when we do git status, we get those file names in untracked files becoz those files are not been tracked by git by using git commit command. The file which was we tracking in below case index.html is already been tracking by git so it will show as changes not staged to commit .



Let say we have 2000 files which we got changed in pwd n out of those only files which have html extension or any other condition like files stating with a or files contain hub, that needs to be staged then we can do it by using below command

git add \*ml or git add a\* or git add \*hub\* 

let say we have 5 changed files in pwd n need to staged only two files,we can do it by adding their names in git add command like below

git add home.html About.html 

* git commit -m “commit\_msg”: once we add the changes to staging area ,final stage is commiting it to repository.that we can do by following command

Git commit -m “commit\_msg”

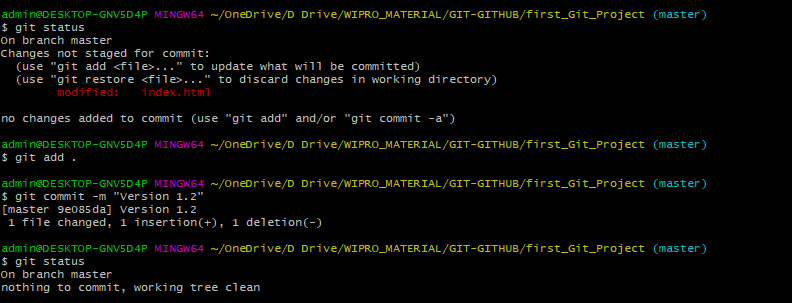
Where -m stands for message

\*If we just write git commit n then press the enter it will open an editor what you selected n configured in config files to write a commit msg .if we close the file without writing a commit msg it will discard the command (git commit)

Commit msg should be in present tensen always should be enough to understand for what purpose changes are made.eg:

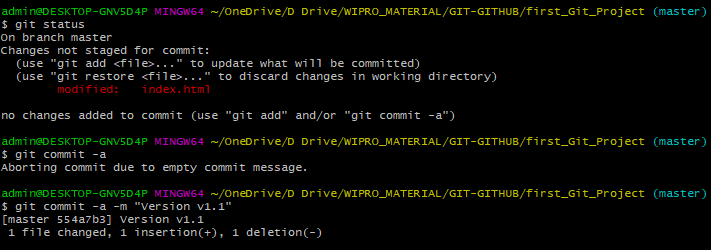
GHI#2345-Fixes bug in a admin login portal

Where GHI stands for Github Issue



Note that without adding a change/set of changes to staging area by using git add . we can directly commit the change/changes to .git repository by using below command

Git commit -a -m “commit\_msg”



When we commit the changes to repository ,git generates hash value(SHA-1 value or 40 character Hex value[0-9,a-f]) based on content of the changed file.for eg:-

490c6101a87af9deeece859c42b4b4bb780f7df6

* git log:this command shows logs of all commits made in the repository ordered by most recent commits.

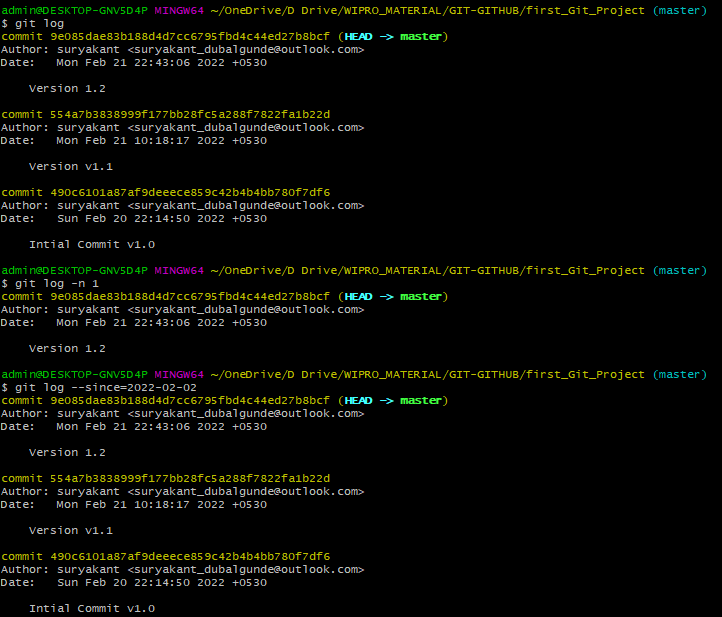
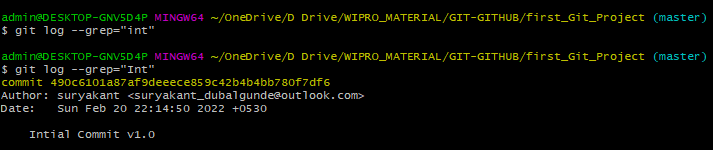
git log -n 2------------------shows logs of 2 most recent commits

git log –since=2021-01-01------shows logs of all commits since the specified date to till today.

Git log –until=2021-01-01------shows logs of all commits untill the specified date.

Git log –author=”surya”--------shows logs of all commits created by specified author.

Git log –grep=”init”--------------shows logs of all commits where commit msg contains specified string value in grep where grep stands for global regular expression.

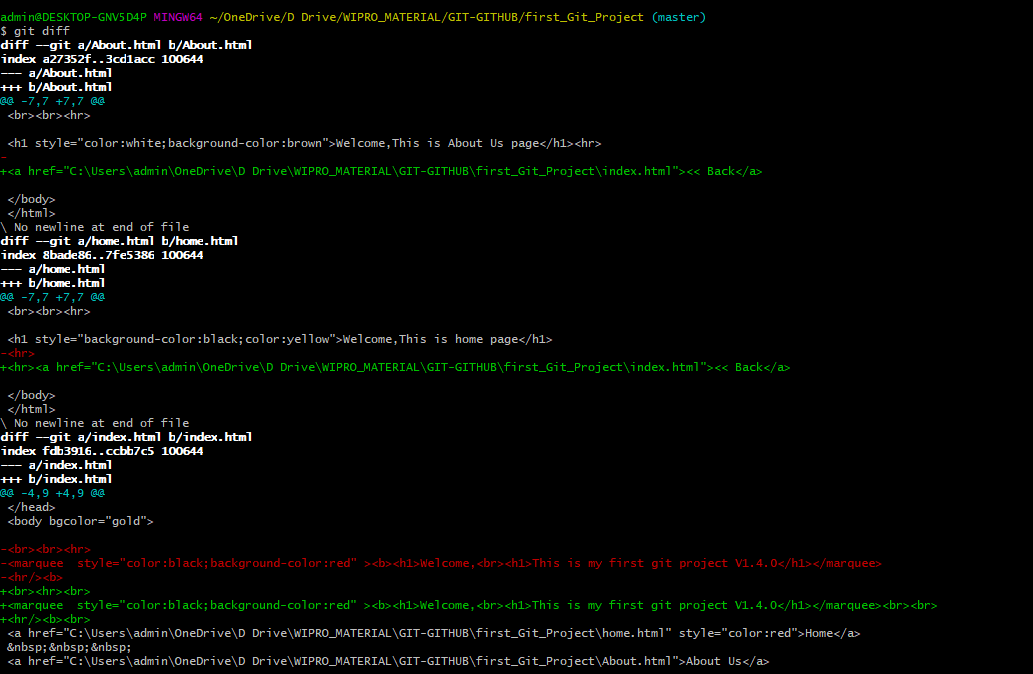
Note that –grep=”” value is case-sensitive.

* Git diff:this command is used to view the changes which we have made in our pwd files.

(++) or green color indicates content that has been added

(--) Or red color indicates content that has been removed

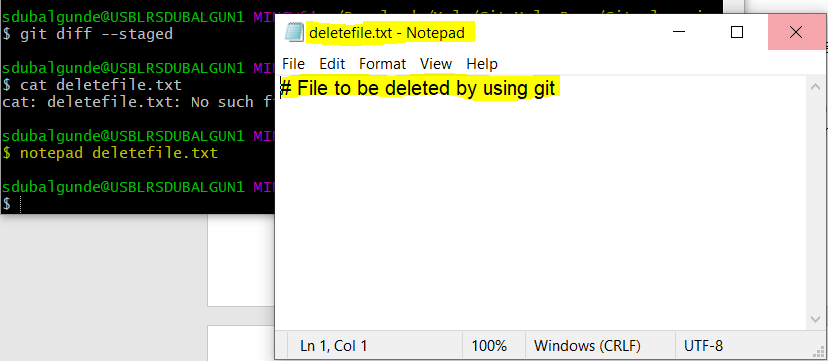
To show the diffences of content in pwd git uses Git diff . git diff command compares pwd file n staging area file copy for this .

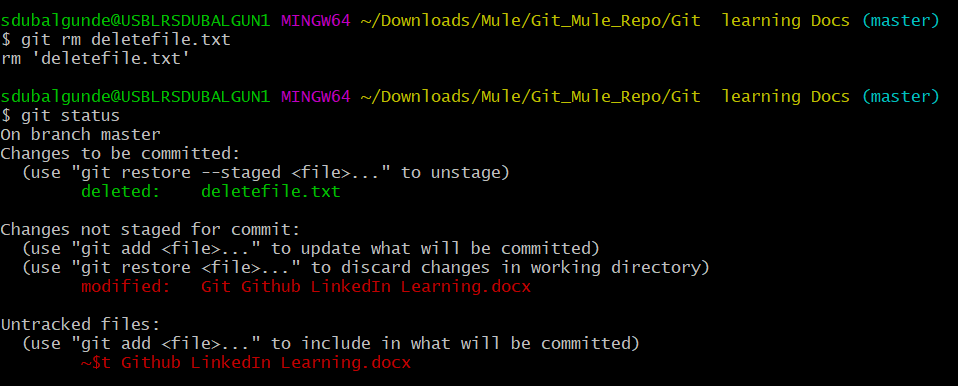


Git diff –staged command is used to see difference between staging file n git repo file..it compares staging area file n git repo file.

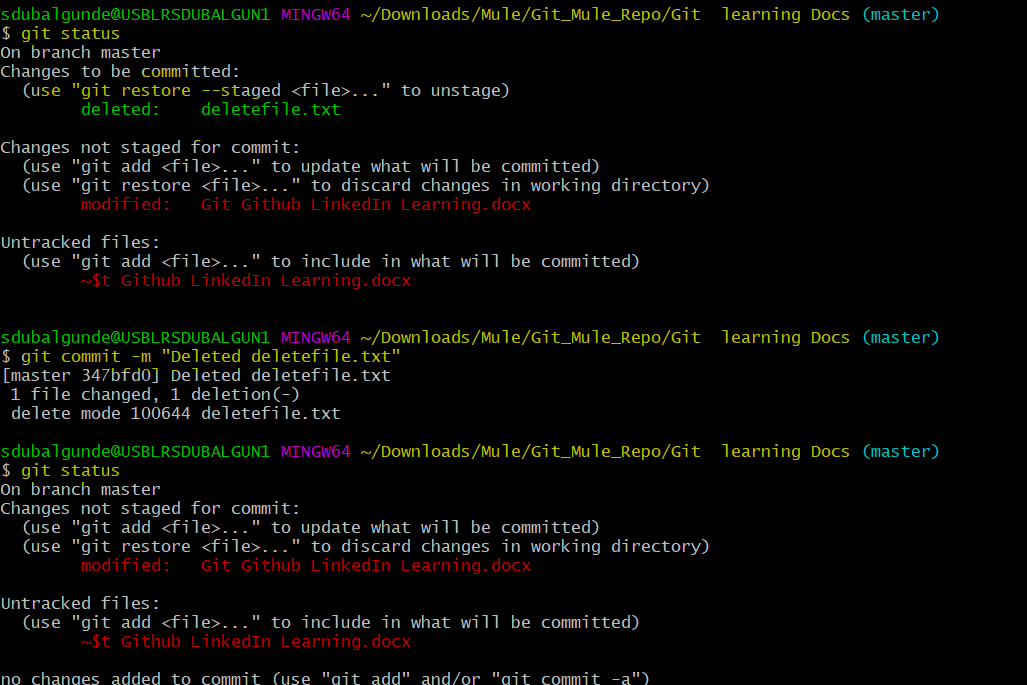
So to compare the changes between pwd file changes n staging area file write git diff

To compare the changes between staging area file n git repo file write git diff –staged.

* Notepad <<filename.ext>>:will open a notepad n will prompt whether the file exists or not,if not then it will create a new file with the filename provided then we can write any comment in the file

* Git rm <file\_to\_be\_removed>:rm command is a unix command which is used to delete particular file. Git rm deletefile.txt will remove the file from git repo and will directly add this delete file change in staging area. 

To add this delete file change in git repo,we need to commit this change by using below cmd:

Git commit -m “commit\_msg” 

* Git restore <<filename>>:This command is used to undo(back to previous changes) the changes from pwd.

Git restore <<filename>> --staged command is used to undo the changes from staging area.