GIT BASH ADVANCE

1. pwd ---------- Present Working Directory path

2. ls ---------- To view directories and files in a folder

3. ls -R ---------- To view subdirectories of directories.

4. ls -t ---------- To view when the files where last modified.

5. ls -l ---------- To view permissions, last modified date, size in bytes of particular folder.

6. ls -lt ---------- To view when the files where last modified with time included in it.

8. ls -la ---------- To view all items including hidden items.

9. ls -lRa ---------- To view the hidden item in the subdirectories which recursively list all the items.

10. ls -lr ---------- To shows all files in reverse order.

11. ls -s ---------- To view directories by its size.

12. ls *.FILE_EXTENSION_NAME ---------- To view all that type of files in that folder.

13. ls Zoo* ---------- To view all the files with "Zoo" in its name.

14. ls .. ---------- list all directories and folder.

15. cd ---------- Go to

16. cd .. ---------- Go to previously directory.

17. cd ../../ ---------- Go back twice.

18. touch ---------- To create a file. {EX- touch a.js}

19. cat ---------- To view what is inside in a file. {EX- cat a.js}

20. cat > a.txt ---------- To write something in a file. ctrl + D to save and exit. ctrl + C to exit.

21. cat >> a.txt ---------- To write more details to the existing file which was cat > a.txt

22. mkdir ---------- Create a directory of name test. {EX- mkdir test}

23. mkdir test && cd test ---------- Create a new directory and go inside that directory.

24. mkdir -p ---------- To create directory inside directory. {EX- mkdir -p frontend/scripts}

25. mv ---------- To move files. {EX- mv script.js runtime_script.js}

26. mv filepath/newname ---------- To rename a file.

27. cp ---------- To copy files {EX- cp filepath new filepath}

28. cp -r ---------- To copy a directory.
29. rm filename ---------- To delete a file.
30. rm -r folderpath ---------- To delete a folder.

PERMISSIONS

+ means adding permissions
- means removing permissions
chmod means change mode
ugo means user, group, others
rwx mean read, write, execute

31. chmod ugo-rwx ---------- To add permission to a file. (what permissions you are adding and to whom like ugo means user, group, others & rwx mean read, write, execute).

31. chmod -R ugo-rwx ---------- To add permission to a folder. (For adding permissions folder -R is required).

32. chmod u+x filename ---------- This will add permissions to execute.

33. chmod g+wx filename ---------- This will add permissions to group to write and execute.

34. chmod u-x filename ---------- This will remove permissions user to execute.

1->x(EXECUTE), 2->w(WRITE), 4->r(READ)

35. chmod 664 folder name ---------- This will give to ugo group like first place 6 is for user(u) second 6 is for group(g) & third 4 is for other(o).

Now here 6 = 4+2 mean 4 is for read and 2 is for write so user(u) will have read & write permissions.

Now here 6 = 4+2 mean 4 is for read and 2 is for write so group(g) will also have read & write permissions.

Now here 4 which is for read and 2 is for write so other(o) will have only write permissions

Now if we want to give all the permissions then the number will be 7(4+2+1; summation of all) for all the cases like - chmod 777 folder name

36. echo 'Hello World'---------- To display a certain message.

37. head filename ---------- View us the first 10 rows of a file.

38. tail filename ---------- View us the last 10 rows of a file.

39. head -20 filename ---------- View the first 20 rows of a file. Same goes with tail.

40. tail -n +25 filename | head -n +5 ---------- To view custom rows. (It will show output starting after 25 and end till 25+5 i.e upto 30)

41. wc filename ---------- To view linecount, wordcount, charactercount of a file.

GREP

42. grep "one" filename ---------- Where "one" has been used in the file.

43. grep "one" filename | wc -l ---------- How many times "one" has been used in the file.

44. grep -c "one" filename ---------- How many times "one" has been used in the file.

45. grep -h "one" filename ---------- Where "one" has been used in the file. (case sensitive)

46. grep -hi "one" filename ---------- where "one" has been used in the file. (not case sensitive)

47. grep -hir "one" directoryname -------- Where "one" has been used in the folder.

48. grep -hin "one" filename ---------- Where "one" has been used in the file inc line numbers. (not case sensitive)

49. grep -hinw "one" filename ---------- Where "one" has been used inside a word also individually. {colone, one, One} (case sensitive)

50. grep -o "one" filename ---------- Only gives us the matched part.

51. grep -w "one" filename ---------- Where "one" has been used in the file.

52. history ---------- To view all the command that i've used.

53. bash filename ---------- This will straightforward execute a Bash script, regardless of the script's execution permissions.

54. grep "ERROR" filename ---------- Will view all the error messages in that file.

55. grep -v "INFO" filename ---------- Will give all the info of the file.

56. grep -A 5 ERROR filename ---------- To view rows after the occurance of ERROR text in a file

56. grep -B 5 ERROR filename ---------- To view rows before the occurance of ERROR text in a file

56. grep -C 5 ERROR filename ---------- To view rows before and after the occurance of ERROR text in a file.

SED

57. sed -n '/ERROR/ p' filename ---------- To print lines with ERROR text.

58. sed 's/ERROR/CRITICAL' filename ---------- Replace ERROR with CRITICAL in the file.

59. sed -ibackup 's/ERROR/CRITICAL/' filename ---- Create a backup of the file.

60. sed '3 s/CRITICAL/VERYCRITICAL/' filename ---- Replace CRITICAL with VERYCRITICAL in line number 3.

60. sed '3,5 s/ERROR/CRITICAL/' filename --------- Replace CRITICAL with VERYCRITICAL in line number 3 to line number 5.

60. sed -n '3,/ERROR/ p' filename ---------- This is used to selectively print a portion of a file, starting from a specific line (line 3 in this case) and continuing until a line containing a specific pattern (in this case, "ERROR") is encountered.

AWK

61. awk '/ERROR/{print $0}' filename ---------- To print lines with ERROR text.

62. awk '{gsub(/ERROR/, "CRITICAL")}{print}' filename--------- Replace ERROR with CRITICAL in the file.

63. awk 'BEGIN {print "LOG SUMMARY\n----------"} {print} END {print "----------\nEND OF LOG SUMMARY"}' filename --------- Add text in the beginning and ending of a file.

64. awk '{print $1, $2}' filename ---------- Print 1st and the 2nd column of the data (file).

65. awk -F "," '{print $1, $2}' filename ---------- Pull a particular category from the data, it will extracts and prints the first two fields of each line.

66. awk '{count[$2]++} END {print count["ERROR]}' filename --- Count the occurance of ERROR in second column of the file.

67. awk '{ if ($1 > 1598863888) {print $0} }' log.txt ------- View the rows after 1598863888 in first column.