public IP address: 3.236.102.150

Alternative if you do not want to create a AWS EC2 instance

===========================================================

Install GIT (Version Control System) on WINDOWS :-

https://git-scm.com › downloads

\* Use GITBASH to practice Linux commands on Windows

Recommended to go the AWS EC2 instance path

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

\* Linux is Unix Operating system

**LIST of which ever commands are learnt**

**=================================**

$> clear

$> env

$> echo $SHELL [ environment variable of Unix ]

/bin/bash [ sh, bash ]

$> echo $LOGNAME

ubuntu

$> whoami

ubuntu

$> id

uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),117(netdev),118(lxd)

$> echo $HOME

/home/ubuntu

$> ls -l [ls -> listing -l long listing ]

$> ls -all [ -all gives you the long listing of all the files / folder, even if it is hidden with "." ]

$> pwd [ present working directory ]

/home/ubuntu

$> cd .. [ change directory to the parent directory of the current directory ]

$> cd - [ change directory with the last cd command ]

$> cd / [ change directory to root directory '/' ]

$> cd home/ubuntu [ change directory from root '/' to ubuntu which is under home ]

$> cd /

$> cd $HOME [ same as above but now change directory using $HOME environment variable ]

NOTE: all the shell/unix commands are in small letter, the options with '-' are also in small letter, the environment variables are

----- always in Capital letter HOME, SHELL, PWD, SSH\_CONNECTION, LOGNAME, USER, PATH

$> echo $SSH\_CONNECTION

$> ssh -i "unixlearning.pem" ubuntu@34.239.150.75 [ this is how you connect externally to a Linux instance externally using the

public key file authentication ]

**VI Editor commands**

**---------------------------**

$> vi <filename>

\* to be in the **INSERT** mode, such that you can starting writing in the editor -> **i**

\* to **UNDO** the last changes use -> **Esc+u**

\* to **INSERT** a new line use -> **o**

\* to be in a ready/**command** mode or to make the INSERT mode off, use -> **Esc**

\* to **Delete** an entire line, use -> **Esc+dd**

\* to repeat the last action -> **.**

\* to **SAVE** the changes in the file, use -> **:w**

\* to **SAVE** as well as **QUIT** the file, use -> **:wq**

\* to **Force** either **SAVE** or **QUIT**, use -> **:w!** or **:wq!** or **:q!**

\* to **REMOVE/DELETE** a character, use -> **x**

\* to **REPLACE** a character, use -> **r**

\* to **REMOVE** a word, use -> **dw**

\* to **APPEND/INSERT** a space -> **a**

\* to **Yank/Copy** an entire line use -> **yy**

\* to **Paste** the Yanked line use -> **p**

\* to **replace** a "word" place the cursor on the first character of the word and type -> **cw**

**How to change the .profile file of the logged in user - so that the prompt can be changed from the default prompt**

**-------------------------------------------------------------------------------------------------------------------------------------------------------**

in the listing if you find files or the folders/directories starting a "." then consider those files/folders as "HIDDEN" files/folders

$> cat .profile

[display the content of the .profile file ]

$> vi .profile

$> PS1='$PWD > '

:wq

----------------------------------------

$> logout [This is to exit the system/session ]

$> ssh .... [login]

$> echo $PWD

$> echo $PS1

**Creating Folders & Files**

**-------------------------------**

$> mkdir my-folder

$> cd my-folder

$> mkdir temp

$> cd temp

$> touch {1..10}.txt

[ to create 10 files with one touch ]

$> rm -rf new-folder

[ to remove the folder with all files - recursively and with force (-rf) ]

There are 3 ways to create a file

1. vi myfirstfile.txt

2. touch mysecondfile.txt [ touch will create a file with 0 bytes ]

3. echo "This is my third file using \"echo\" command" > mythirdfile.txt

[ this creates a file with the line inserted, using 'single re-direction' ]

4. echo "I am writing the second line using \"echo\" command" >> mythirdfile.txt

[ this appends to the end of the file with the text using 'double re-direction']

**Copy**

**-------**

$> cp mysecondfile.txt myfourthfile.txt

$> cat mythirdfile.txt > myfourthfile.txt

**Grep**

**-------**

$> cat mythirdfile.txt | grep "NOT"

$> cat mythirdfile.txt | grep "NOT" > myfourthfile.txt

**Remove**

**------------**

$> rm myfourthfile.txt [ deletes the file ]

$> rm -f myfourthfile.txt [ force delete the file ]

$> rm -r <folder-name>

$> touch myfirstfile-{1..10}.txt

[ This creates 10 files by the name myfirstfile-1.txt, myfirstfile-2.txt … myfirstfile-10.txt ]

$> rm myfirstfile\*

[ To remove / delete the files with the name starting with myfirstfile, the rest of the filename is replaced by \* ]

$> touch myfile-{1..10}.txt

$> rm -rf ./myfolder

[ to remove (rm) the folder recursively (-r) as well as remove all the files inside the recursive folders (-f) ]

**Miscellaneous commands**

**-----------------------------------**

$/home/ubuntu > df .

[ df - report file system disk space usage - here we are trying to see the disk utilization of the current (.) folder ]

$> man df

[ man stands for Unix Manual for the command ]

$> logout

[ to logout of the current user which is logged in ]

$/home/ubuntu > sudo su -l

[ sudo -> is a super user privilege given to the use ubuntu. Here we are using the sudo privilege of ubuntu to login to "**root**" use which is all powerful "super user su" ]

$/root > su -l ubuntu

[ from root you have logged into ubuntu ]

$/home/ubuntu > logout

[ ubuntu -> root -> ubuntu => logout -> root -> logout -> ubuntu -> logout -> exit from the system ]

$> ps -eaf

[ It will list all the "**processes**" that are running in the background of the Unix Server - it is like the **Task Manager** from Windows ]

$> ps -eaf | grep sshd

[ out of all the Unix processes running in the background we are filtering (grep) out the "sshd" processes ]

$> ps -eaf | head -10

[ This will list the first 10 lines from the process list ]

$> ps -eaf | tail -20

[ This will list the last 20 lines from the process list ]

$> mv onefile ./myfolder

[ Moving a file "onefile" from the current folder to a folder "myfolder" - so the file "onefile" will no longer be available under the current folder as it has moved out to myfolder ]

$> history

[ will produce the history of the commands that were issued till the start of the session - One can also use the history keys "Up" and "Down" arrow keys to retrieve the history of commands]

$> find . -name onefile -print

[ Use find command to search for a file or folder or device from the current folder to all the recursive folders ]

$> find . -name one\* -print

[ Use the find command with partial matching file/folder/device name that is using a pattern or wild-card (\*) search ]

$> find . -name one?i\* -print

[ Using the pattern replacement with ? and wild-card (\*) search ]

$> which vi

[ This command gives if the utility has been installed and the executable is in which folder - here vi has been installed in /usr/bin/ ]

$> top

[ gives a view of the processes running in the background and their CPU and Memory usages ]

**LAB**

**====**

1. Display your current directory.

2. Change to the /etc directory.

3. Now change to your home directory using only three key presses.

4. Change to the /boot/grub2 directory using only eleven key presses.

5. Go to the parent directory of the current directory.

6. Go to the root directory.

7. List the contents of the root directory.

8. List a long listing of the root directory.

9. Stay where you are, and list the contents of /etc.

10. Stay where you are, and list the contents of /bin and /sbin.

11. Stay where you are, and list the contents of ~.

12. List all the files (including hidden files) in your home directory.

13. List the files in /boot in a human readable format.

14. Create a directory testdir in your home directory.

15. Change to the /etc directory, stay here and create a directory newdir in your home directory.

16. Create in one command the directories ~/dir1/dir2/dir3 (dir3 is a subdirectory from dir2, and dir2 is a subdirectory from dir1 ).

17. Remove the directory testdir.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**USER Management in Linux**

**=======================**

The local user database in Linux is /etc/passwd directory.

$> tail etc/passwd

[ Starting from the left columns denotes username, an x, user id, primary group id, a description, name of home directory and a login shell. ]

root user:

The root user is the superuser and have all the powers for creating a user, deleting a user and can even login with the other user's account. The root user always has userid 0.

$> head -1 /etc/passwd

Adding user:

$> useradd -m -d /home/debasis -c "debasis" debasis

[ we have created a user**debasis** along with creating a home directory (-m), setting the name of home directory (-d), and a description (-c). ]

$> tail -10 etc/passwd

/etc/default/useradd:

File /etc/default/useradd contains some user default options. The command useradd -D can be used to display this file.

$> useradd -D

userdel:

To delete a user account userdel command is used.

$> userdel -r debasis

[ removing the user home directory of the user along with the user account ]

$> tail -10 etc/passwd

Password management for the user created:

$> sudo su -l debasis

$> passwd

OR

$> sudo passwd debasis

chage:

The chage command can be used by a user to know the information about their password. The -l option is used to list the information.

$> chage -l debasis

**GREP (Pattern Searching ):**

**======================**

$> vi file1.txt

---------------------------------------

This is Debasis

Welcome to Debasis's Unix session

learn Linux

Linux is very easy & interesting

---------------------------------------

$> grep "D" file1.txt

$> grep l file1.txt

$> grep L file1.txt

$> grep "v" file1.txt

*concatenating:*

$> grep "Deb" file1.txt

$> grep in file1.txt

$> grep is file1.txt

*expression -> one or other:*

$> grep -E 'D|g' file1.txt

$> grep -P 'D|g' file1.txt

$> grep -G 'D|g' file1.txt

$> grep -G 'D\|g' file1.txt

*Match The End Of A String:*

$> vi duplicate.txt

-----------------------------

one

four

six

eight

four

nine

ten

one

-----------------------------

$> grep r$ duplicate.txt

$> grep e$ duplicate.txt

Match The Start Of A String:

$> grep ^o duplicate.txt

*Separating Words:*

$> vi file2.txt

-----------------------------

someone just called me

some people are worst

I need to buy some mangoes

-----------------------------

$> grep some file2.txt

$> grep '\bsome\b' file2.txt

$> grep -w some file2.txt

*Using Grep options:*

$> vi file3.txt

-------------------------------------------------------------------------------------------------------

unix is great os. unix is opensource. unix is free os.

learn operating system.

Unix linux which one you choose.

uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.

-------------------------------------------------------------------------------------------------------

$> grep -i "UNix" file3.txt

$> grep -c "unix" file3.txt

$> grep -l "unix" \*

$> grep -l "unix" file1.txt duplicate.txt file2.txt file3.txt

$> grep -w "unix" file3.txt

$> grep -o "unix" file3.txt

$> grep -n "unix" file3.txt

$> grep -v "unix" file3.txt

$> grep "^unix" file3.txt

$> grep "os$" file3.txt

**SED (Stream editing ):**

**===================**

* SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution).
* SED command in unix supports regular expression which allows it perform complex pattern matching.

$> vi file4.txt

-------------------------------

today is Thursday

today is not a holiday

-------------------------------

$> sed -i 's/today/tomorrow/' file4.txt

*Simple Back Referencing:*

Double ampersand is used to search and find the specified string. It will print the found string with sed command.

$> echo fourty | sed 's/four/&&/'

*A Dot For Any Character:*

In regex a simple dot can signify any character

$> echo xxxx-xx-xx | sed 's/....-..-../YYYY-MM-DD/'

*Multiple Back Referencing:*

When more than one pair of parenthesis is used it is called grouping. Here each of them can be referenced separately as three consecutive numbers.

$> echo 2014-06-30 | sed 's/\(....\)-\(..\)-\(..\)/\1:\2:\3/'

$> echo 2014-06-30 | sed 's/\(....\)-\(..\)-\(..\)/\1\_\2\_\3/'

$> echo 2014-06-30 | sed 's/\(....\)-\(..\)-\(..\)/\2:\3:\1/'

$> echo 2014-06-30 | sed 's/\(....\)-\(..\)-\(..\)/\3:\2:\1/'

*White Space:*

The white space syntax is '\s' and tab space syntax is '\t'.

$> echo -e  'this\tis\tjavatpoint'

$> echo -e  'this\tis\tjavatpoint' | sed 's/\s/ /g'

*Optional Occurrence:*

You can specify something optional by specifying it with **(?)** question mark.

$> vi file5.txt

------------------------------------------------------------------

hi

hii

hiii

hiiii

hiiiii

--------------------------------------------------------------------

$> cat file5.txt | sed 's/iii\?/Y/'

[ we have made third 'i' as optional. It means that two 'i' are must to be converted into 'Y'. ]

*Exact n Times Occurrence:*

Exact times occurence is specified by **"{times}"**.

$> cat file5.txt | sed 's/i\{3\}/Y/'

*Occurence In Range:*

We can specify occurence in terms of range also. For example, if we'll specify range as {m,n}, then 'm' denotes minimum times occurence and 'n' denotes maximum times occurence.

$> cat file5.txt | sed 's/i\{3,4\}/Y/'

*Replacing or substituting string :*

$> sed 's/unix/linux/' file3.txt

*Replacing the nth occurrence of a pattern in a line :*

$> sed 's/unix/linux/2' file3.txt

*Replacing all the occurrence of the pattern in a line :*

$> sed 's/unix/linux/g' file3.txt

*Replacing from nth occurrence to all occurrences in a line :*

$> sed 's/unix/linux/3g' file3.txt

*Parenthesize first character of each word :*

$> echo "Welcome To The Geek Stuff" | sed 's/\(\b[A-Z]\)/\(\1\)/g'

**AWK command in Unix/Linux with examples**

**=====================================**

Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan

Awk is a scripting language used for manipulating data and generating reports.

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements

that define text patterns that are to be searched for in each line of a document and the action that is to be taken

when a match is found within a line.

**Create the following text file for AWK manipulation**

$> cat > employee.txt

ajay manager account 45000

sunil clerk account 25000

varun manager sales 50000

amit manager account 47000

tarun peon sales 15000

deepak clerk sales 23000

sunil peon sales 13000

satvik director purchase 80000

--- *Default behavior of Awk, with no pattern matching*

$> awk '{print}' employee.txt

--- *Print the lines which match the given pattern*.

$> awk '/manager/ {print}' employee.txt

--- *Splitting a Line Into Fields ($0 represents the whole line)*

$> awk '{print $1,$4}' employee.txt

--- *To return the second column/item from employee.txt:*

$> awk '{print $2}' employee.txt

**---** *Built-In Variables In Awk ->* ***NR, NF, FS***

**NR** -> display line nos.

$> awk '{print NR,$0}' employee.txt

**NF** -> no. of fields, built-in variables (Display Last Field)

$> awk '{print $1,$NF}' employee.txt

Display Line From 3 to 6

$> awk 'NR==3, NR==6 {print NR,$0}' employee.txt