Basics of Linux command & Shell Scripting

Note: In Linux there is no GUI. So, we have to perform all the tasks using command line.

pwd – to see present working directory

ls – to list all the directory i.e files and folders in particular directory

cd – to change the present working directory

And if want to go back from pwd then instead of give complete path we can do this

cd .. – move back to one directory

cd ../.. – move back to two directory

cd ubuntu/bundle – if we want to go two directories in more / depth we can do like this.

ls -ltr – to list all the files and folders with the timestamps, who is the owner of the file, who other owns this file, when it is created and other information like whether it is a file or directory, file size, permissions etc. we use this command. (list file and folders with its properties)

ls -al – use to list all the file and folders with all information

mkdir – to create a new directory (folder) eg: mkdir test

touch – to create a file only and is used most in automation.

(suppose we have to create a 1000s of file then we can't use vi/vim. I have to use touch only. If we create 1000s file and keep it open then it makes other process slow and we need to close all these files to execute the process properly and if someone is using that file then OS will tell that it is being used by someone so unnecessary we can't keep the file open).

vi – to create a file and also opens the file to write inside it *(available by default in any linux machine*)

vim – to create a file and also opens the file to write inside it *(we need to install it in linux machine as it is not available by default*)

but to write content on it we have to use **escape (esc)** button to tell linux that I have to write on the file and then click **i** to get into insert mode and then now you can write on to it. And press esc to save it using **:** followed by **wq** for saving the file and followed by **!** and then press enter.

:wq! – to save the file (w -> write, q -> quit)

Cat – to print the content of the file (concatenate the content of file into the command shell).

rm – to remove the file

rm -r – to remove a directory eg: rm -r test

rm -rf - used to forcefully remove files and directories recursively

cp – use to copy the file or folder and past it on the relative path or folder

Eg: cp test.sh Desktop or cp test.sh ..

Kill – to kill a processes

Kill -9 1111(process id or process name) – (-9 instructs the linux compiler or linux kernel to kill this process id)

COMMAND TO CHECK THE CPU, MACHINE AND ITS PERFORMANCE COMPLETELY

free / free -g – to see the memory of the server (look for RAM health)
nproc – to see number of CPUs allocated (look for CPU health)

df / df -h - to see disk size (total, used and available, %used) on particular machine.

And we can see and manage all these three at one place using the top command to see all the processes running and how much CPU they are utilizing.

top – to get the complete information of memory, cpu and disk i.e we can monitor everything using one place.

How you monitor the CPU node health?

Ans : Using top command or I can run the custom shell scripts. And the basic parameters that I use to evaluate the node health is CPU and RAM.

More About Shell Scripting

- We use a man (manual) as a suffix before any command to get the manual / details of that command.
- Eg – *man ls or man touch etc.*
- We use **.sh** extension to save shell scripting
- Eg – *touch first-shell-script.sh*
- We use *double click* to copy the content, file or folder name and using ctrl + v or cmd + v depending on the OS to paste it on terminal.
- #!/ - shebang (it is a specific syntax or indentation)

  Shebang followed by path and shell they are using *eg: /bin/bash or /bin/dash or /bin/sh or /bin/ksh*

Note: *these bash, dash, sh, ksh are different executables of Linux depends on the Linux machine we are using and there is slight syntax difference between them.*
bash is most popular and widely used*.*
So if you are using ubuntu machine use ***#!/bin/bash***

Earlier both *#!/bin/sh* and *#!/bin/bash* are same due to linking concept but now it is not the same because some if the OS have decided to use dash as default so our script might not execute if it is written in bash scripting where dash is default.

echo – use to print the statement in shell scripting similar to print in python and cout in c++.

Eg: echo "Hello I am Anurag"

sh – to execute the shell scripting file

Eg: sh test.sh

./ - use to execute any type of executable file like .py, .cpp etc.

Eg: ./test_python.py

REMEMBER: In linux suppose you have created the file and you are the owner of the file but you don't have the permissions to execute it due to security reason. By default, you can only read and write on that file.
But you can change the permission of that file using chmod command.

chmod – to grant permissions to a file (i.e change the mode)

Eg: chmod 744 test.sh

There are 3 categories to whom permission should be given

1. user (owner)
2. group
3. everyone (other)

And it follows number to represent the permission

4 – read only

2 – write only

1 – execute only

And the three categories are represented like this

User	**group**	**everyone**
7	7	7

Here all there thee have all the three permissions of read, write and execute.

User	**group**	**everyone**
7	6	4

Here user has all three permission (r, w, e) and group has only two permissions (4+2) i.e (r, w) and others (everyone) has only one permission of read only (4).
And according we can modify this number to whom we have to give what access

Let's see how to change the permission

chmod permission file_name

Eg: chmod 444 test.sh or chmod 744 test.sh or chmod 764

Or there is another way to do same is by using this

chmod u+x test.sh – here I have given permission to user to execute this file.

chmod g+w test.sh – here I have given permission to group to write into this file.

chmod o+w test.sh – here I have given permission to other to write into this file.

---

history – it gives the history of all the command that we have used

# - used for commenting the line inside shell script.

Example of shell script (suppose we are inside some folder or directory)

Use command vi sample-shell-script.sh to create and open the file

```
#!/bin/bash

#create a folder test

mkdir test

#move inside this test directory

cd test

#create two files inside it

touch firstFile secondFile
```

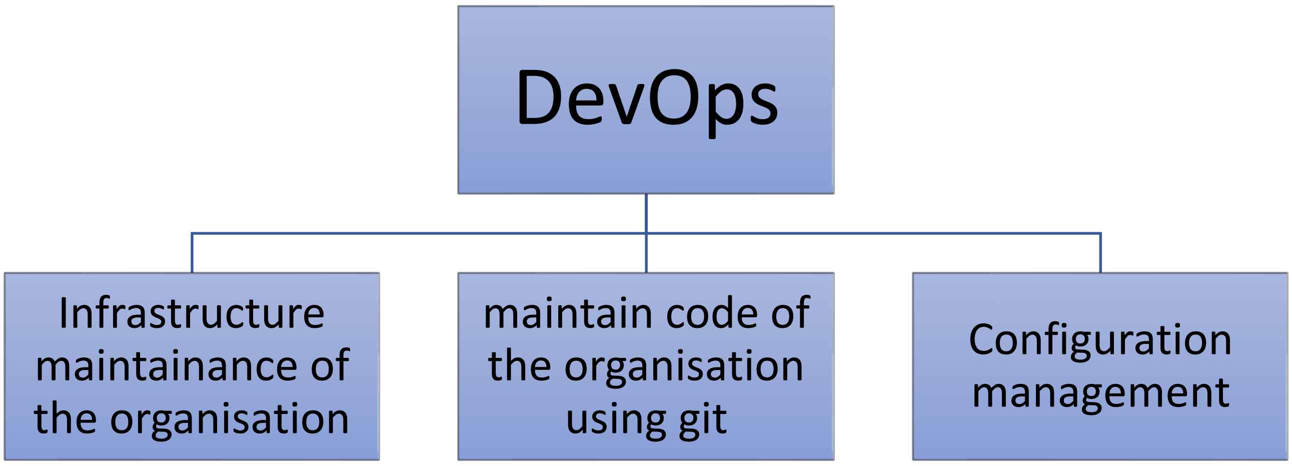
Press esc then :wq! to save the file

Now use command chmod 744 to change the permission. We can keep it as 777 also as for now we don't have any security concern.

Then use ./sample-shell-script.sh to execute the file.

Shell Scripting for DevOps

Why we need shell scripting?


DevOps
Infrastructure maintainance of the organisation
maintain code of the organisation using git
Configuration management

For all these purposes on day-to-day basis we use shell scripting.

Note: There are tools like Ansible for configuration management then why we need shell scripting. Because some/my organizations don't use ansible or using shell scripting I get more insights to monitor the machine health that I don't from the ansible that's why I prefer shell scripting.

Set -x – run the command in the debug mode (actually it prints each command before giving the output which helps DevOps engineer to debug things easily)

```
    #####################

    # Author: Anurag

    # Date: 05/12/2025
```

```
#
# This script outputs the node health
#
# Version: v1
####################
set -x //executes the script in debug mode (-x)
df -h
free -g
nproc
```

Note: we can also write an echo statement before each command but for a very huge scripts always use set -x and can also use the combination of both as per the requirement. But only writing an echo statement for a huge script is not feasible.

- Always write meta data information
- Whenever you start a script use this set -x to set the script in debug mode

And if you don’t want the user to know what command command you are using then just simply comment it out.

Ps – this will print only the running process but not the background process

ps -ef - Command to print all the processes running on the VM.

- **-e**: Selects all processes running on the system, not just those associated with the current terminal.
- **-f**: Provides a full-format listing for each process, including additional details.

The ps -ef command is a fundamental tool for:

- **Process Management**: Viewing and understanding all running processes.
- **Troubleshooting**: Identifying processes consuming high CPU or memory.
- **System Monitoring**: Gaining insights into resource usage and application activity.

ps -ef | grep “amazon” – it will it will only list the processes that is only running by amazon.

This grep only fetches the information that is required.

| - this is called pipe parameter. Its task is to send the output of one command to the other as an input here we can see we have two command one is ps -ef and another is grep “amazon”.

Eg: ./test.sh | grep “1” – this will only print the echo statement where it finds one.

Note: on vm we generally need the process ID to kill, trip dump or anything we want to do with the process we use process Id.

What will be the output of this command date | echo “date is”

Expectations: date is Fri Dec 5 06:12:55 UTC 2025 (because first command pass the output to second command).

Output: date is

Answer : Because date is a default shell command and what it does it sends the output to stdin.

Note: In any system there are three channels or three log flows i.e stdin, stdout and stderr.

So, what happens here is the date command sends the output to stdin i.e to input one.

Final Answer: It will only print “date is” because date is a system default command and what this command does is it sends the output to stdin but pipe will not be able to receive the information from stdin. Pipe can only receive the information if the command is not sending the information to stdin and if the command is ready to pass the information to the next command.

grep – it prints the line that match special patterns.

Eg: ps -ef | grep nginx

Output:

```
root       3776       1  0 Dec03 ?       00:00:00 nginx: master process /usr/sbin/nginx -g
daemon on; master_process on;

www-data    3778    3776  0 Dec03 ?       00:00:00 nginx: worker process

www-data    3779    3776  0 Dec03 ?       00:00:00 nginx: worker process

ubuntu     58404   58163  0 13:14 pts/0    00:00:00 grep --color=auto nginx
```

awk – pattern scanning and process language what it does is it can filter out the information from the output.

Note: So the difference between grep and awk command is the grep command will give the entire statement but the awk command is very powerful it filters out the statement or sentences and gives the specific column.

Command: ps -ef | grep “nginx” | awk -F” ” ‘{print $2}’

Output:

```
3776
3778
3779
58411
```

Just the id of the process.

Here -F” ” is a field separator and here the field separator is space.

Eg:

Command: vi test

```
My name is Anurag.
I work in Microsoft.
```

Esc -> :wq! -> enter

Command: grep name test

Output : My name is Anurag.

Command: grep name test | awk -F” ” ‘{print $4}’

Output: Anurag

Note: Whenever you use this command make sure you retrieve the right information i.e always make sure you use right column number and use grip at the right place.

Important: Whenever using pipe make sure to use a specific syntax called set -e and set -o

set -e – exit the script when there is an error except error from pipe (it will not catch pipe failure).

set -o – pipe fail then this will exit and if this command is not present then the script will look for last command after the pipe. If it does not have an error then execute further and if it finds error then it gets exit.

Eg:

Sdfgh | echo | fghjk

With set -o then it get exit

And without set -o it will exit

Sdfgh | echo | fghjk | echo

With set -o then it get exit while executing the first command as it is an invaid command.

And without set -o it will not exit as it only consider the last command and that is a valid command.

Note: So, whenever you use set -e then also include set -o in the script.

Sometimes people used to write like this
set -exo pipefail

It is the same command. But never do this as by writing separately we have an option to remove it. So instead of this write like this only

```
        set -x #debug mode

        set -e #exit the script when there is an error

        set -o pipefail
```

Another major use case of DevOps engineer is supposed if there are 100 applications running and if one of the applications is failing then first thing that come to the mind is log file. So, every time you run into an error then there is your one success sutra that is go to your log file and find errors there.

Generally, log files have huge information and as a DevOps engineer you want to understand what are the errors in the log file and we have to do all this in the linux platform.

There are different levels of log file i.e TRACE, INFO, ERROR and we have to find the logs that are at the error level.

Cat logfile | grep error -> we can simply use cat logfile to print the log and use grep error to filter the error from that log file.

But there is a twist here whenever these logs get appended like there are more and more logs so what people usually do is they upload the logs to some storage platform say google storage service or amazon storage service like S3 or Azure blob. So, at the end of the day these log files are stored in some external storage devices and is not present on the virtual machine. So how we retrieve these information is using the curl command. We just use the curl command and provide the url of the log file location.

curl – it retrieves the information from the internet. Use to create API call or any external device call from internet or download anything.

Eg: curl https://dummyurl.com/logfile | grep ERROR

Eg: curl -X GET api.foo.com #making an get API call

wget – the non-interactive network downloader. It can work on background and what wget does is that if we give the url of logfile then it download it instead of printing the log on the consol i.e it stores the information.

And we have to run grep command on the downloaded file.

So I have to run the command twice to perform the same task so instead of using wget here we are using curl.

find – this command is used to find the file or anything on machine using file path

Eg:

find / - it means find everything means all the files and folders in the VM.

sudo su – : this command will take you to the root user (sudo means substitute user do or superuser do).

su – using this command we can swith to another user

Eg: su anurag or su xyz

Note: not everybody has access to the root user and root user has some privilege over other user. So, use use sudo command to get the sudo privilages and run the command here as another user also. i.e is we run some command on behalf of the root user.

Eg:

find / -name pam – this will not allow to access all the files and send a message permission denied. But

sudo find / -name pam – now we can access those files.

If, if else and for loop

```
If [expression]

then

        Write whatever you have to do

else

        Write whatever you have to do

fi    #it tells if loop ends here.
```

Eg:

vi if-else.sh

```
a=4

b=10

if [$a > $b]

then

        echo “a is greater than b”

else

        echo “b is greater than a”

fi
```

For Loop

```
For i in {1,100};

        do echo $1;

done


```

trap – it is used for trapping the signals

Note: kill -9 1111, while executing this command there is a signal passed to the Linux saying okay so this person is asking to kill a specific file so this is a signal and another thing lets say there is a script that is getting executed and what we usually do is to terminate the script use ctrl + c. So when pressing it on a keyboard how does a Linux machine should know while we press ctrl + c it has to stop the execution of a script.

So when we press any key from a key board a signal is emitted that this Linux machine has received. And accordingly, it performs the action.

And there are lots of signal available on the linux machine like

- SIGINT is a signal generated when a user presses Control-C. This will terminate the program from the terminal.
- SIGALRM is generated when the timer set by the alarm function goes off.
- SIGABRT is generated when a process executes the abort function.

- SIGSTOP tells LINUX to pause a process to be resumed later.
- SIGCONT tells LINUX to resume the processed paused earlier.
- SIGSEGV is sent to a process when it has a segmentation fault.
- SIGKILL is sent to a process to cause it to terminate at once.

So the trap command in Linux is a shell built-in that allows you to define actions to be taken when a shell script receives a signal or when the script exits. This is particularly useful for handling unexpected interruptions or ensuring proper cleanup operations.

- trap can intercept various signals sent to a process, such as SIGINT (Ctrl+C), SIGTERM (termination signal), SIGHUP (hangup signal), and others.
- When a specified signal is received, trap executes a defined set of commands or a function instead of the default action for that signal.
- This enables scripts to gracefully handle interruptions, perform cleanup, or even ignore certain signals.

Basically, if we have a trap mechanism that is set on my machine and even if they press ctrl + c don't do anything or if they press ctrl + c send me a notification on an email notification using SMTP server or just print an echo statement saying that you can't use the ctrl + c because the owner of the linux file or linux machine said that does not want to execute ctrl + c. Such thing can be used by the trap command.

So trap signal is basically something that traps the signals and performs the execution on your behalf.

Eg: syntax: trap message/whatever task you have to perform name of the signal

trap "echo don't use the ctrl+c" SiGINT

Suppose there is a database and our shell script is running to populate the database items and half of the content is printed and someone came to your terminal and pressed ctrl+c and only half of the information is populated and user thinks that some information is there and this is the information I can start using. So instead of this what we can do is whenever somebody press ctrl+c delete every content.

trap "rm -rf *" SIGINT

Write a script to print the nos. from 1 to 100 which are divisible by 3 and 5 but not with 15.

```
#!/bin/bash

####################
```

```
	# Author: Anurag
	# Date: 05/12/2025
	#
	# This script outputs the number divisible by 3 and 5 but not 15 between 1 to 100
	#
	# Version: v1
	####################
	for i in {1..100}; do
	if (( (i % 3 == 0 || i % 5 == 0) && i % 15 != 0 )); then
		echo $i
	fi;
	done
```

Write a script to print the count of ‘s’ present in the word mississipi

```
	#!/bin/bash
	####################
	# Author: Anurag
	# Date: 05/12/2025
	#
	# This script outputs the s count in word mississipi
	#
	# Version: v1
	####################
	x= mississipi
	grep -o “s” <<<”$x” | wc -l
```

-o – here stands for only i.e only get the output of single letter s.

wc -l – here stands for word count length

<<< - here we are redirecting the output of a standard input i.e x to the grep command

PROJECT 1

```
    #!/bin/bash
    ####################
    # Author: Anurag
    # Date: 09/12/2025
    #
    # This script will report the AWS resource usage
    #
    # Version: v1


    #run the script in debug mode
    set -x


    #list S3 buckets
    echo “Print the list of S3 buckets”
    aws s3 ls


    #This command runs the aws s3 ls command and overwrites the file
    named resourceTracker with its output
    echo “Print the list of S3 buckets and store the output in resourceTracker”
    aws s3 ls > resourceTracker


    #list EC2 Instances
    echo “Print the list of EC2 instances”
    aws ec2 describe-instances
```

```
# get the particular Instance id from a JSON then we have use jq for json parser and
there is yq for YAML parser also

echo “get the particular Instance id from a JSON”

aws ec2 describe-instances | jq ‘Reservations[].Instances[].InstanceId’


#list lambda

echo “Print the list of lambda functions”

#This command runs the aws lambda list-functions command and appends its output
to the end of the existing resourceTracker file

aws lambda list-functions >> resourceTracker


#list IAM users

echo “Print the list of IAM users”

aws iam list-users
```