28|3|22

Linux. [LSS DAY-01]

Q What is Git and function related.

→ Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

→ Linus Torvalds created Git in 2005 for the development of the Linux kernel.

{OR}

st Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development

- Git is used to tracking changes in the source code
- The distributed version control tool is used for source code management.
- It allows multiple developers to work together

* Features of Git *

1. Tracks history
2. Free and open source
3. Supports non-linear development.
4. Create backups,
5. Scalable
6. Supports collaboration
7. Branching is easier
8. Distributed development.

---

* Programming Constructs.

* **Wildcards :-** There are some characters that are evaluated by the shell in a special way. They are called shell metacharacters or "wildcards".
These characters are neither numbers or letters. For ex → the * ? and [ ] are used for filename expansion. The <, >, 2>, >>, and | symbol are used for standard I/O redirection and pipes. To prevent these characters from being interpreted by the shell, they must be quoted.

E.g → rm *, ls ??; cat file [1-3]
echo "Hello World"

* **Displaying output :-**

To print output to the screen, the echo command is used. Wildcards must be escaped with either a backslash or matching quote
E.g :- echo "Hello World"

* Local Variable → Local variable are in scope, ~~the echo command~~ for the current shell. When a script ends they are no longer available; i.e., they go out of scope Local variable can also be defined with the built-in declare function. Local variables are set and assigned values

Ex → variable_name = value
declare variable_name = Value

name = "Vipul Tiwari"
x = 5

* Global Variable → Global variables are called environment variables and are created with the export-built in command. They are set for the currently running shell and any process spawned from that shell. They go out of scope when the script ends

→ The built-in declare function with the -x option also sets an environment variable and marks it for export

Ex → export VARIABLE_NAME = value
declare -x Variable_NAME = value
export PATH =/bin:/usr/bin:

* Extracting values from Variable ⇒ To extract values from variables, a dollar sign is used

Ex → echo $variable_name
echo $ name
echo $ PATH

* Reading user input ⇒ The user will be asked to enter input. The read command is used to accept a line of input. Multiple arguments to read will cause a line to be broken into words, and each word will be assigned to the name variable.

Ex → echo "what is your name"
read name
read name1 name2 ...

* Arithmetic operators → The Bash shells support integer arithmetic. The declare -i command will declare an integer type variable. The Korn shell's typeset command can also be use for backward compatibility. Integer arithmetic can be performed on variables declared this way. Otherwise the (( )) (let command) syntax is used for arithmetic operations

Ex → declare -i variable_name used for bash
typeset -i variable_name can be used to be compatible with ksh
((n=5+5))
echo $n

* Arguments → Arguments can be passed to a script from the command line. Positional parameters are used to receive their values from within the script

Ex → scriptname arg1 arg2 arg3 ...
In a script
echo $1 $2 $3 - Positional parameters
echo $* - All the positional "
echo $# - The no. of positional parameters

* The shbang line :- The "sh bang" line is very first line of the script and tells the kernel know what shell will be interpreting the lines in the script. The sh bang line consists of a #! followed by the full pathname to the shell, and can be followed by options to control the behaviour of the shell.

Ex :- #!/bin/bash