DevOps

Docker

Containers	Virtual M/c

We can run any Application on Docker or create an Image of that Application using
- Dockerfiles

Bash - born against shell

from History we can run the command again just by typing !no. ex: !3 to

get that command.

Package manager

Ubuntu – Apt – Advance package tool

- apt list ← list of packages

if no packages found first update

① apt update

- apt install nano.
- Ctrl+L → shortcut to clear window

pwd – print working directory

cd ~ – return to Home directory

Ctrl+W → To remove entire word

touch → can be used to create multiple files

- To rename file

→ mv oldfilename.txt new.txt

- To remove files of same pattern

remove file*

→ will remove all files of that pattern.

remove directory

rm -r directory/

View and Edit file

- view – cat filename.txt
  → if file is small

cat means → concatinate

- More filename.txt
  IF large file.

* Improved version of more is less
  apt install less

We can also use head | tail cmd

ex. head -n 5 /path

viewing | edit

- nano
- cat
- more
- less

- head
- tail

Searching

- grep → case sensetive

grep -i → non-case sensetive

↳ Global regular expression print.

You can Also do grep -i Hell* File*
to actually check everything

To search recursively we use -r

ls -a → To see Hidden files
find → To see the all files and directory.

find -type f -iname "f*"

finding files in root whoes extention is .py

find -type f -name "*.py" > pythonresults.txt

Chaining command

;

Can be chained using semicolon

→ If one command fails other commands still going to run and this is where && (and) operator helps to stop other command execution.

&& – and operator (if 1 fails stop)

|| – or operator (if one cmd fails run others)

* Piping |

single line = "|" (pipe)

Break-up long commands using \

• Enviornmental variables

Users variables can be written into

• .bashrc Hidden file from Home (ls -a)

To see hidden files

• Redirect and Append

To redirect we use > which we overwrite all content from the file and write new content

- To Append >>

— Enviornmental Variables is used to store information into Docker
for ex. You can write DB_user in the Docker file so that, that can store into container and we dont have to remember

.bashrc ← Is the file where you can save env
- To view current enviornments you can use printenv command
- We can also use echo $PATH
or echo $DB_USER
$ symbol states its env variable.

Once you write changes to bashrc thoes are visible only to new session → Basically after restart in windows term because bashrc executes at start.
— we can reload session or we can use source command to reload bashrc
ex. source .bashrc

Users

legacy → useradd → also adduser ← new

usermod

userdel

Whenever we create a new user using useradd -m Sam
By default it creates shell and it is limited that's why we use Bash. To get Bash By default We can use -s

ex. usermod -s /bin/bash

Passwords are stored in shadow file
/etc/shadow

To login as root user in Docker
docker exec -it -u Sam bash
$ ← $ states that you are not root user
# ← states that you are root user.

Groups

groupadd
groupmod
groupdel

groups

Primary group

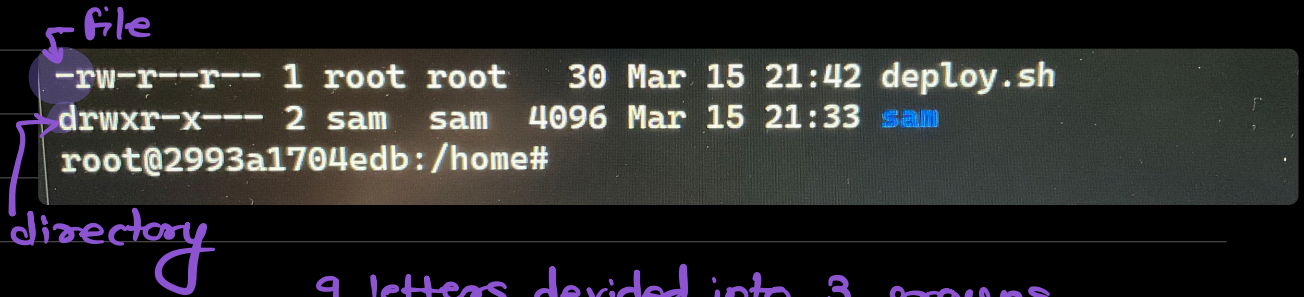
4 secondary groups

groups sam — To show groups of sam

• File permissions.

view permissions

ls -l



9 letters devided into 3 groups

rwx rwx rwx

↓ ↓ ↓

for user for group everyone else.

chmod

To add / remove permissions

chmod u+x (add, execution permission, user)

u - user — x - execute

g - group — w - write

o - other — r - read

ex.2 chmod o+x filename

ex3 chmod og+r+w-x filename.sh