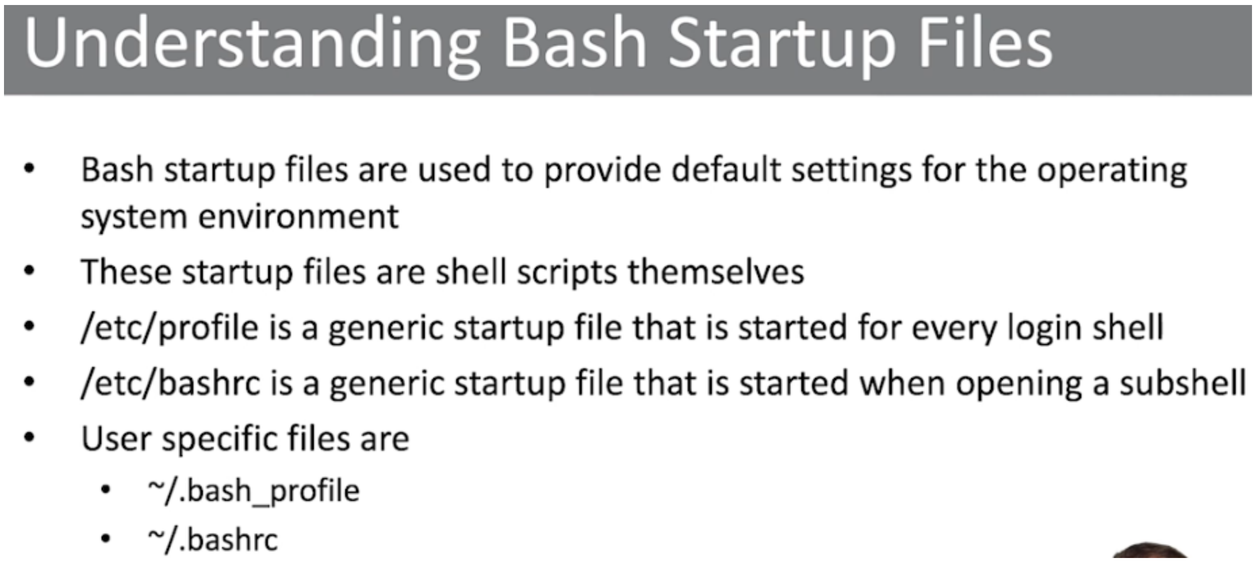
Note 1: In Some OS **.bash_profile** Called As **.profile**

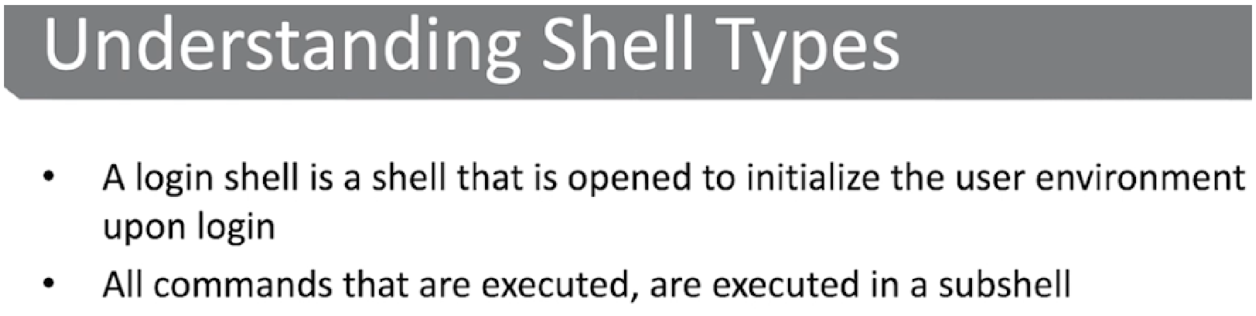
Note 2: In Some OS **.bashrc** Called As **.login**



Understanding Bash Startup Files

- Bash startup files are used to provide default settings for the operating system environment
- These startup files are shell scripts themselves
- /etc/profile is a generic startup file that is started for every login shell
- /etc/bashrc is a generic startup file that is started when opening a subshell
- User specific files are
  - ~/.bash_profile
  - ~/.bashrc

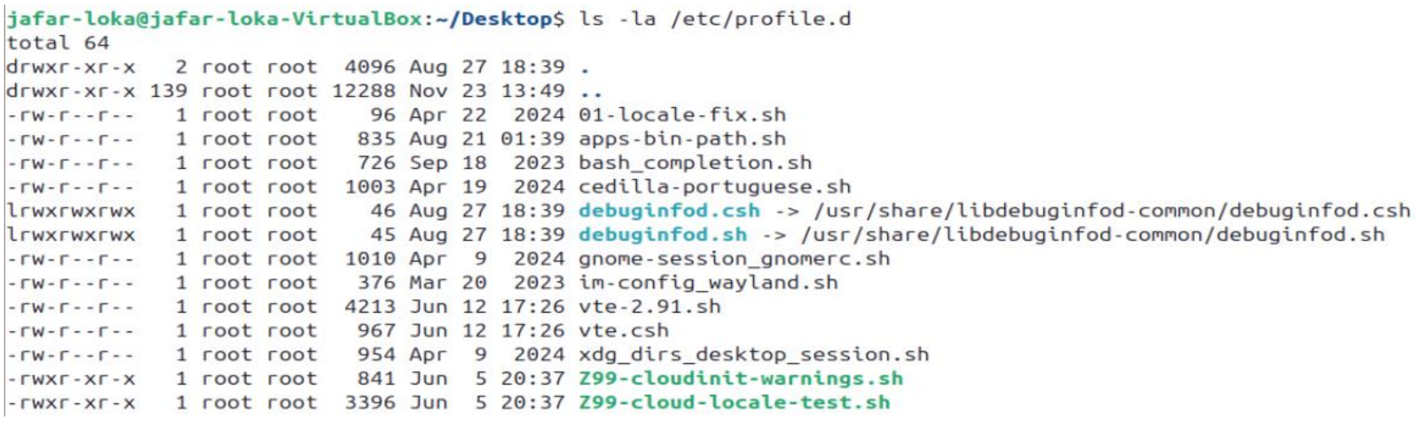
****



Understanding Shell Types

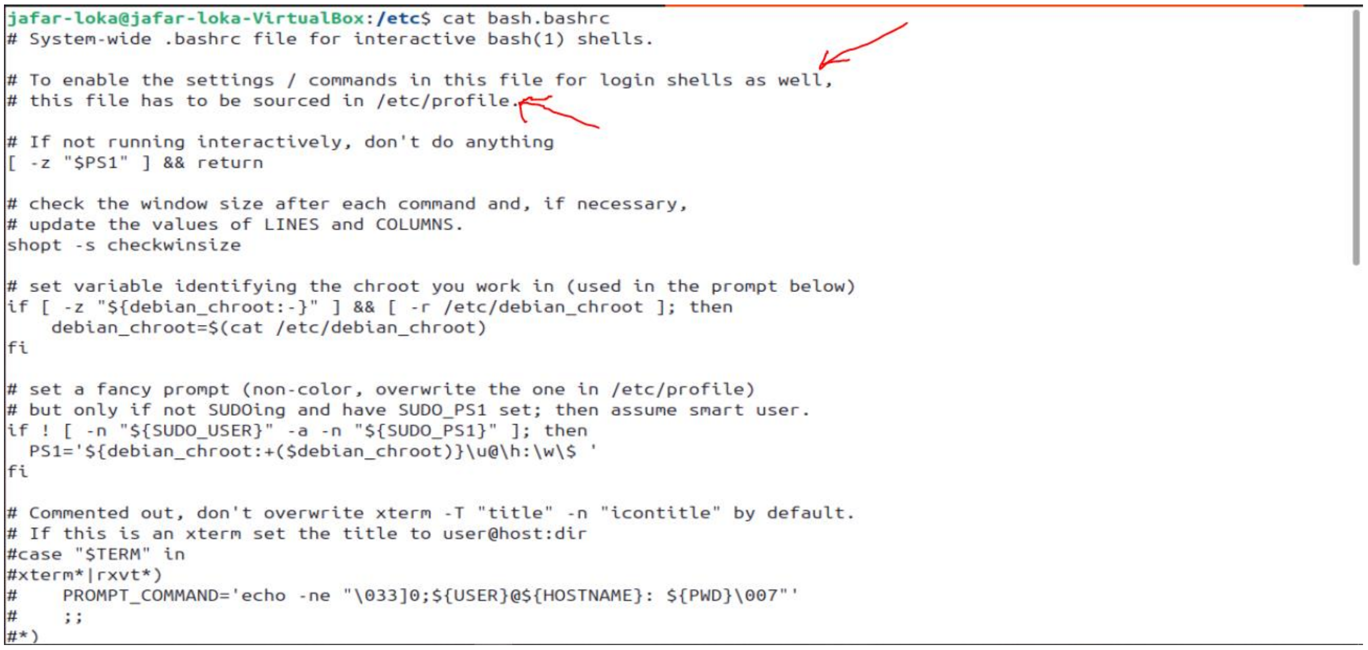
- A login shell is a shell that is opened to initialize the user environment upon login
- All commands that are executed, are executed in a subshell

****



```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ ls -la /etc/profile.d
total 64
drwxr-xr-x   2 root root  4096 Aug 27 18:39 .
drwxr-xr-x 139 root root 12288 Nov 23 13:49 ..
-rw-r--r--   1 root root    96 Apr 22  2024 01-locale-fix.sh
-rw-r--r--   1 root root   835 Aug 21 01:39 apps-bin-path.sh
-rw-r--r--   1 root root   726 Sep 18  2023 bash_completion.sh
-rw-r--r--   1 root root  1003 Apr 19  2024 cedilla-portuguese.sh
lrwxrwxrwx   1 root root    46 Aug 27 18:39 debuginfod.csh -> /usr/share/libdebuginfod-common/debuginfod.csh
lrwxrwxrwx   1 root root    45 Aug 27 18:39 debuginfod.sh -> /usr/share/libdebuginfod-common/debuginfod.sh
-rw-r--r--   1 root root  1010 Apr  9  2024 gnome-session_gnomerc.sh
-rw-r--r--   1 root root   376 Mar 20  2023 im-config_wayland.sh
-rw-r--r--   1 root root  4213 Jun 12 17:26 vte-2.91.sh
-rw-r--r--   1 root root   967 Jun 12 17:26 vte.csh
-rw-r--r--   1 root root   954 Apr  9  2024 xdg_dirs_desktop_session.sh
-rwxr-xr-x   1 root root   841 Jun  5 20:37 Z99-cloudinit-warnings.sh
-rwxr-xr-x   1 root root  3396 Jun  5 20:37 Z99-cloud-locale-test.sh
```

****



```
jafar-loka@jafar-loka-VirtualBox:/etc$ cat bash.bashrc
# System-wide .bashrc file for interactive bash(1) shells.

# To enable the settings / commands in this file for login shells as well,
# this file has to be sourced in /etc/profile.

# If not running interactively, don't do anything
[ -z "$PS1" ] && return

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# set variable identifying the chroot you work in (used in the prompt below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi

# set a fancy prompt (non-color, overwrite the one in /etc/profile)
# but only if not SUDOing and have SUDO_PS1 set; then assume smart user.
if ! [ -n "${SUDO_USER}" -a -n "${SUDO_PS1}" ]; then
  PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi

# Commented out, don't overwrite xterm -T "title" -n "icontitle" by default.
# If this is an xterm set the title to user@host:dir
#case "$TERM" in
#xterm*|rxvt*)
#    PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
#    ;;
#*)
```

****

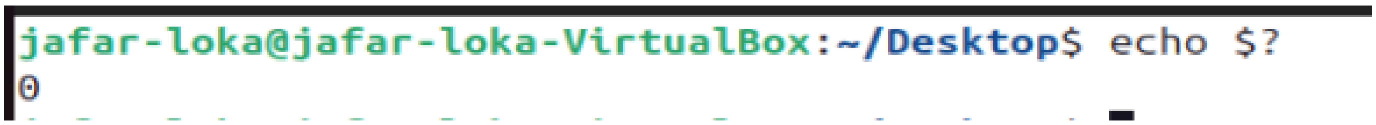
To Learn About What Files Are Executed When The User Login/Logout We Can Learn About .bashrc And .bashrc_login && ...etc

****

Understanding Exit Codes

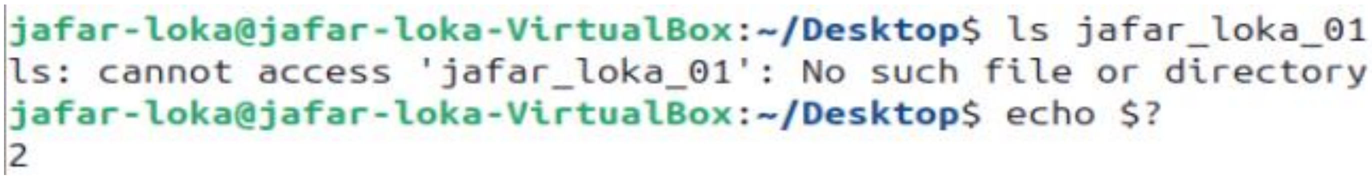
- After execution, a command generates an exit code
- The last exit code generated can be requested using **echo $?**
- If 0, the command was executed successfully
- If 1, there was a generic error
- The developer of a program can decide to code other exit codes as well
- In shell scripts, this is done by using **exit *n*** in case an error condition occurs

****



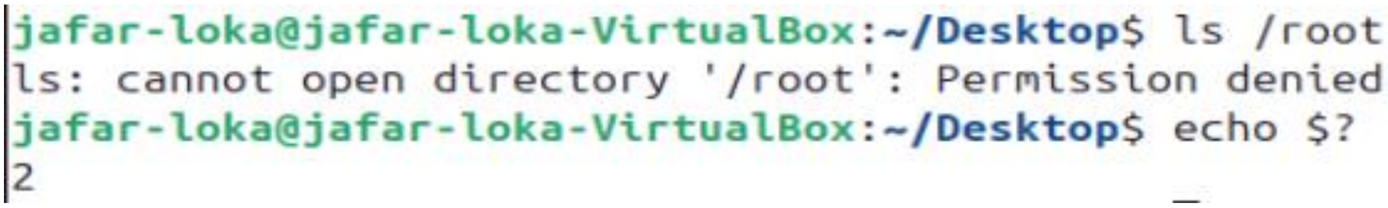
```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ echo $?
0
```

****



```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ ls jafar_loka_01
ls: cannot access 'jafar_loka_01': No such file or directory
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ echo $?
2
```

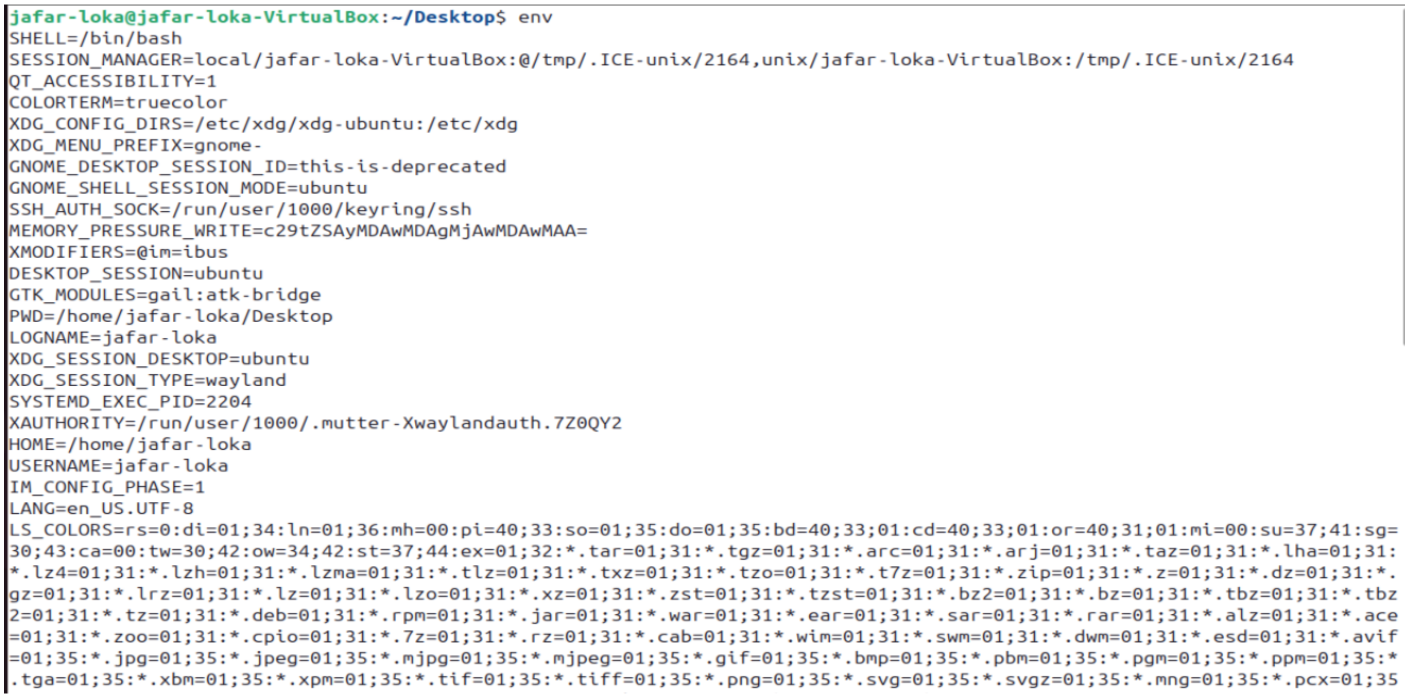
****



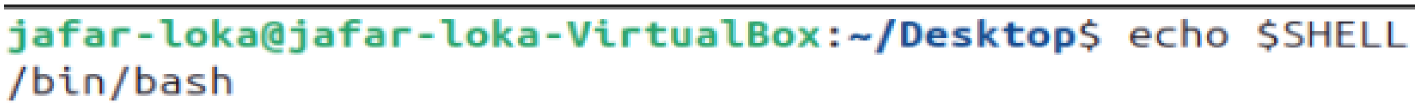
```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ ls /root
ls: cannot open directory '/root': Permission denied
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ echo $?
2
```

****

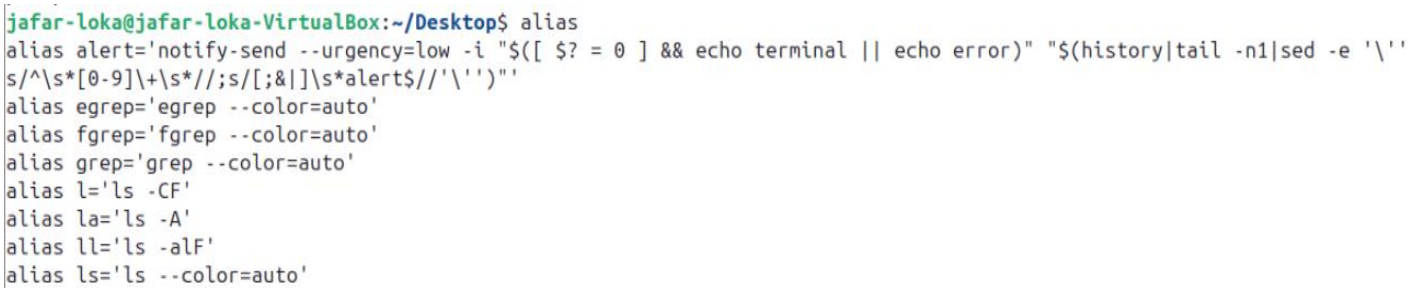
Note (To Remember):



```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ env
SHELL=/bin/bash
SESSION_MANAGER=local/jafar-loka-VirtualBox:@/tmp/.ICE-unix/2164,unix/jafar-loka-VirtualBox:/tmp/.ICE-unix/2164
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
MEMORY_PRESSURE_WRITE=c29tZSAyMDAwMDAgMjAwMDAwMAA=
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
GTK_MODULES=gail:atk-bridge
PWD=/home/jafar-loka/Desktop
LOGNAME=jafar-loka
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=wayland
SYSTEMD_EXEC_PID=2204
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.7Z0QY2
HOME=/home/jafar-loka
USERNAME=jafar-loka
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=
30;43:ca=00:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:
*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.
gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz
2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace
=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif
=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*
.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35
```

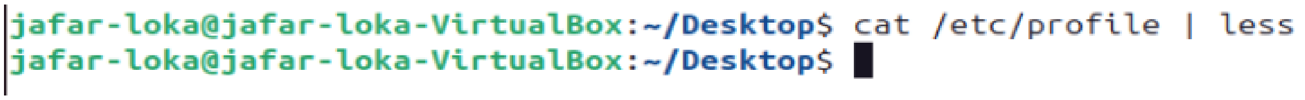


```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ echo $SHELL
/bin/bash
```



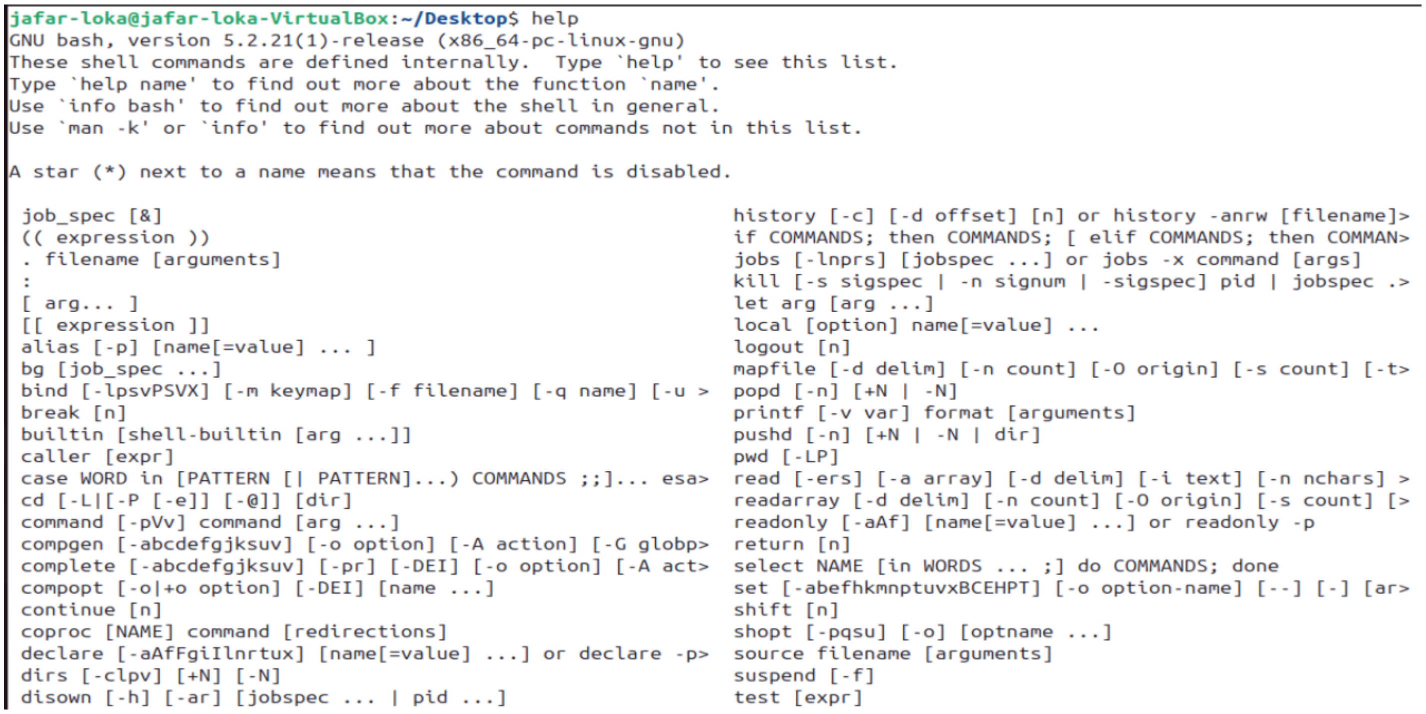
```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''
s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
```

****



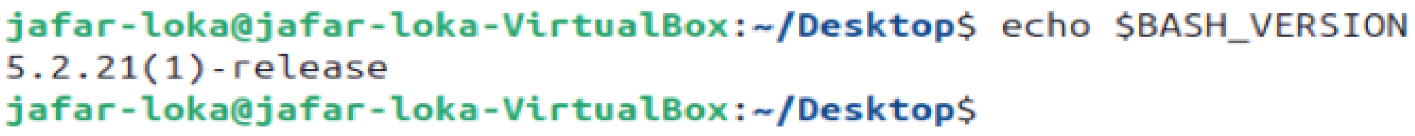
```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ cat /etc/profile | less
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ █
```

****



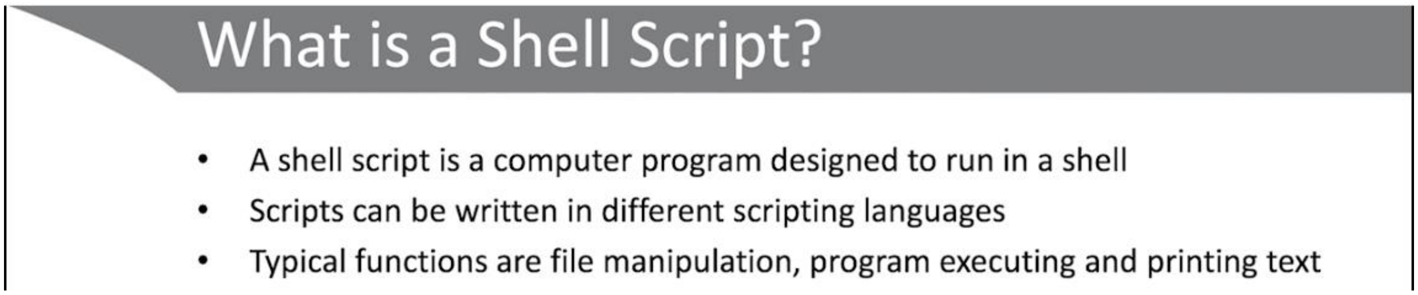
```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ help
GNU bash, version 5.2.21(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 job_spec [&]                                                  history [-c] [-d offset] [n] or history -anrw [filename]>
 (( expression ))                                              if COMMANDS; then COMMANDS; [ elif COMMANDS; then COMMAN>
 . filename [arguments]                                        jobs [-lnprs] [jobspec ...] or jobs -x command [args]
 :                                                             kill [-s sigspec | -n signum | -sigspec] pid | jobspec .>
 [ arg... ]                                                    let arg [arg ...]
 [[ expression ]]                                              local [option] name[=value] ...
 alias [-p] [name[=value] ... ]                                logout [n]
 bg [job_spec ...]                                             mapfile [-d delim] [-n count] [-O origin] [-s count] [-t>
 bind [-lpsvPSVX] [-m keymap] [-f filename] [-q name] [-u >    popd [-n] [+N | -N]
 break [n]                                                     printf [-v var] format [arguments]
 builtin [shell-builtin [arg ...]]                             pushd [-n] [+N | -N | dir]
 caller [expr]                                                 pwd [-LP]
 case WORD in [PATTERN [| PATTERN]...) COMMANDS ;;]... esa>    read [-ers] [-a array] [-d delim] [-i text] [-n nchars] >
 cd [-L|[-P [-e]] [-@]] [dir]                                  readarray [-d delim] [-n count] [-O origin] [-s count] [>
 command [-pVv] command [arg ...]                              readonly [-aAf] [name[=value] ...] or readonly -p
 compgen [-abcdefgjksuv] [-o option] [-A action] [-G globp>    return [n]
 complete [-abcdefgjksuv] [-pr] [-DEI] [-o option] [-A act>    select NAME [in WORDS ... ;] do COMMANDS; done
 compopt [-o|+o option] [-DEI] [name ...]                      set [-abefhkmnptuvxBCEHPT] [-o option-name] [--] [-] [ar>
 continue [n]                                                  shift [n]
 coproc [NAME] command [redirections]                          shopt [-pqsu] [-o] [optname ...]
 declare [-aAfFgiIlnrtux] [name[=value] ...] or declare -p>    source filename [arguments]
 dirs [-clpv] [+N] [-N]                                        suspend [-f]
 disown [-h] [-ar] [jobspec ... | pid ...]                     test [expr]
```

****



```
jafar-loka@jafar-loka-VirtualBox:~/Desktop$ echo $BASH_VERSION
5.2.21(1)-release
jafar-loka@jafar-loka-VirtualBox:~/Desktop$
```

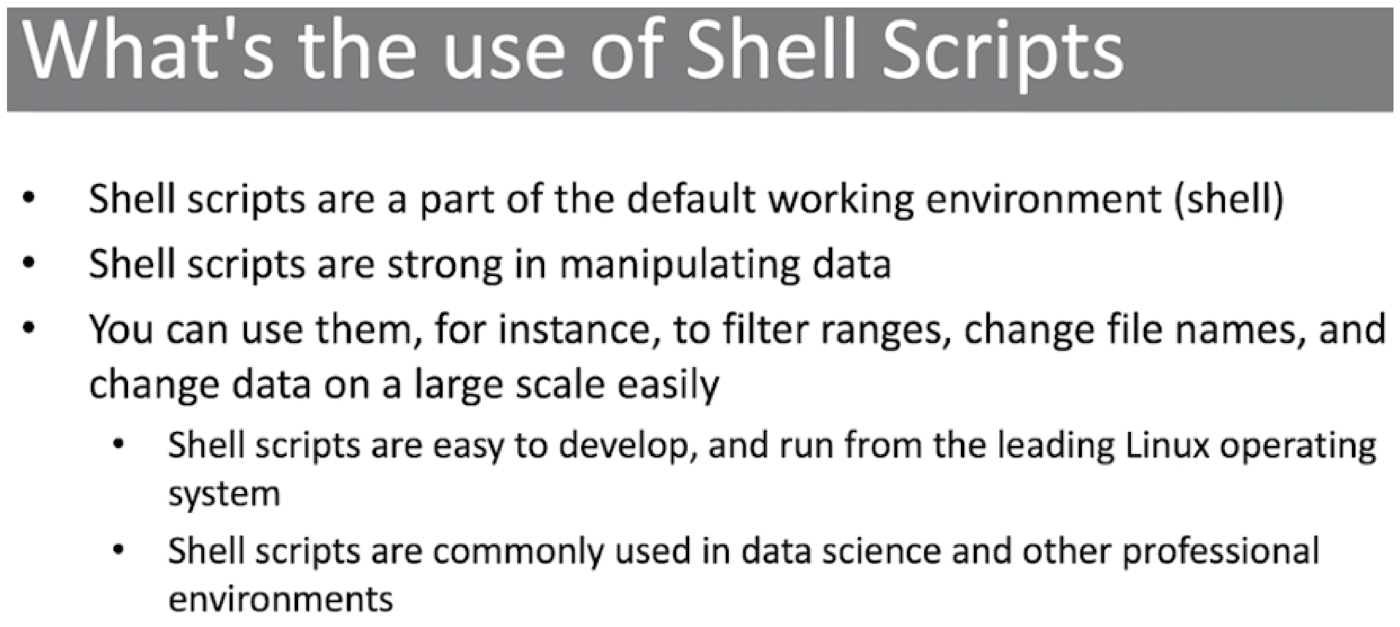
****



What is a Shell Script?

- A shell script is a computer program designed to run in a shell
- Scripts can be written in different scripting languages
- Typical functions are file manipulation, program executing and printing text

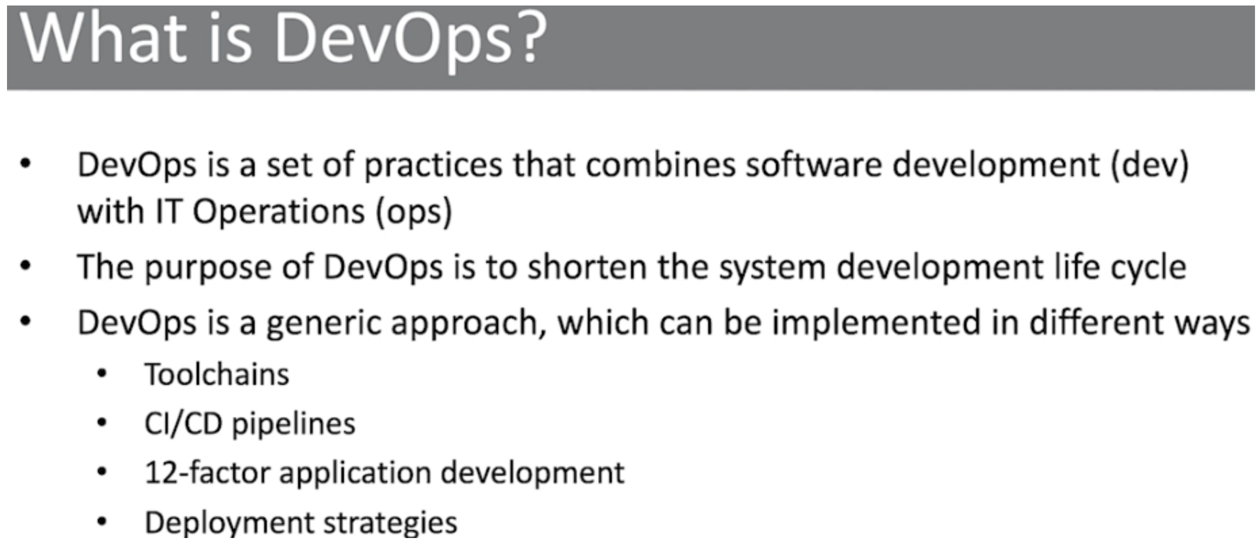
****



What's the use of Shell Scripts

- Shell scripts are a part of the default working environment (shell)
- Shell scripts are strong in manipulating data
- You can use them, for instance, to filter ranges, change file names, and change data on a large scale easily
  - Shell scripts are easy to develop, and run from the leading Linux operating system
  - Shell scripts are commonly used in data science and other professional environments

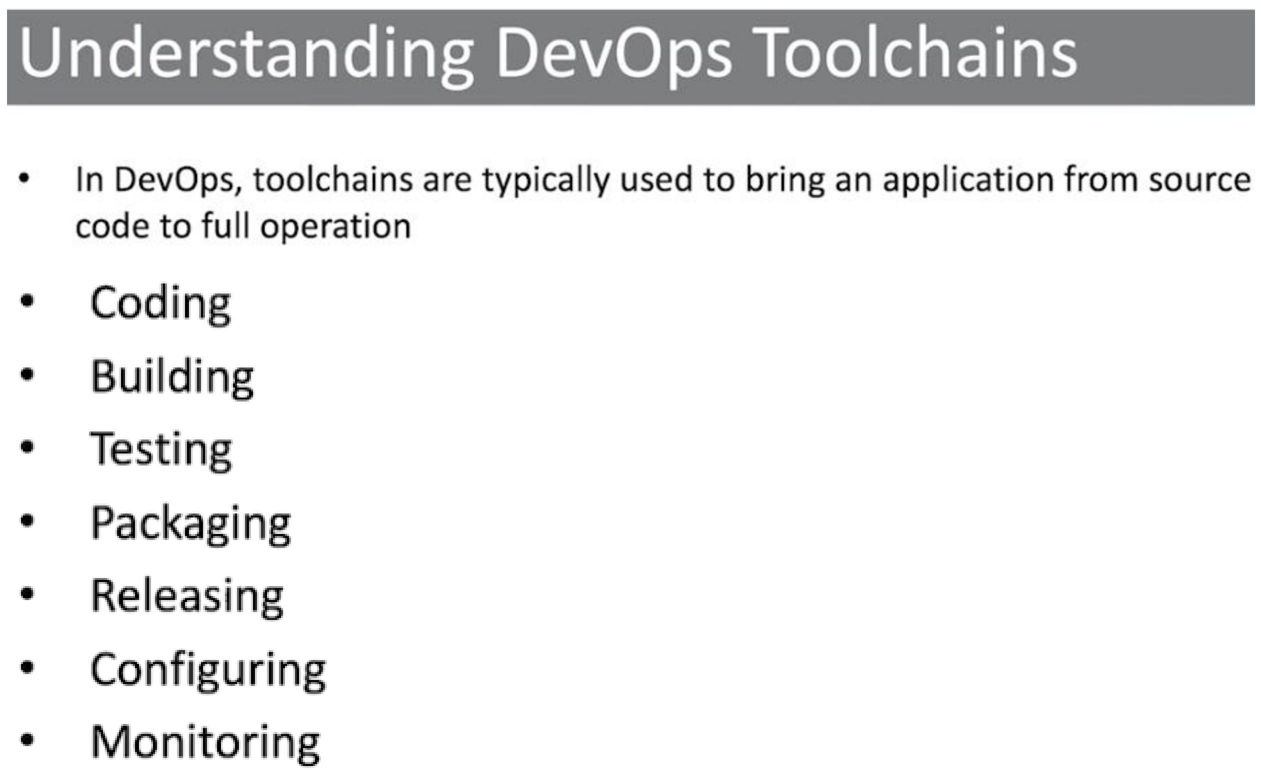
****



What is DevOps?

- DevOps is a set of practices that combines software development (dev) with IT Operations (ops)
- The purpose of DevOps is to shorten the system development life cycle
- DevOps is a generic approach, which can be implemented in different ways
  - Toolchains
  - CI/CD pipelines
  - 12-factor application development
  - Deployment strategies

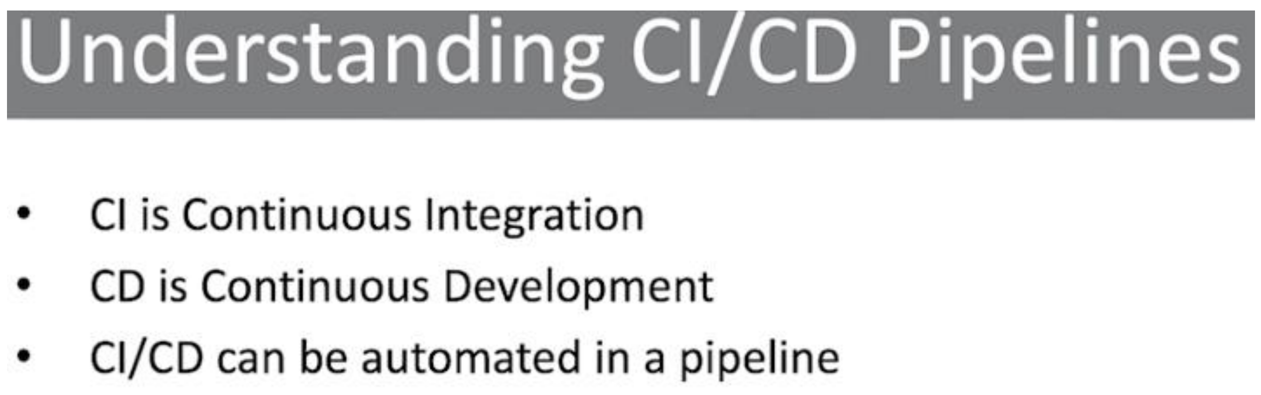
****



Understanding DevOps Toolchains

- In DevOps, toolchains are typically used to bring an application from source code to full operation
- Coding
- Building
- Testing
- Packaging
- Releasing
- Configuring
- Monitoring

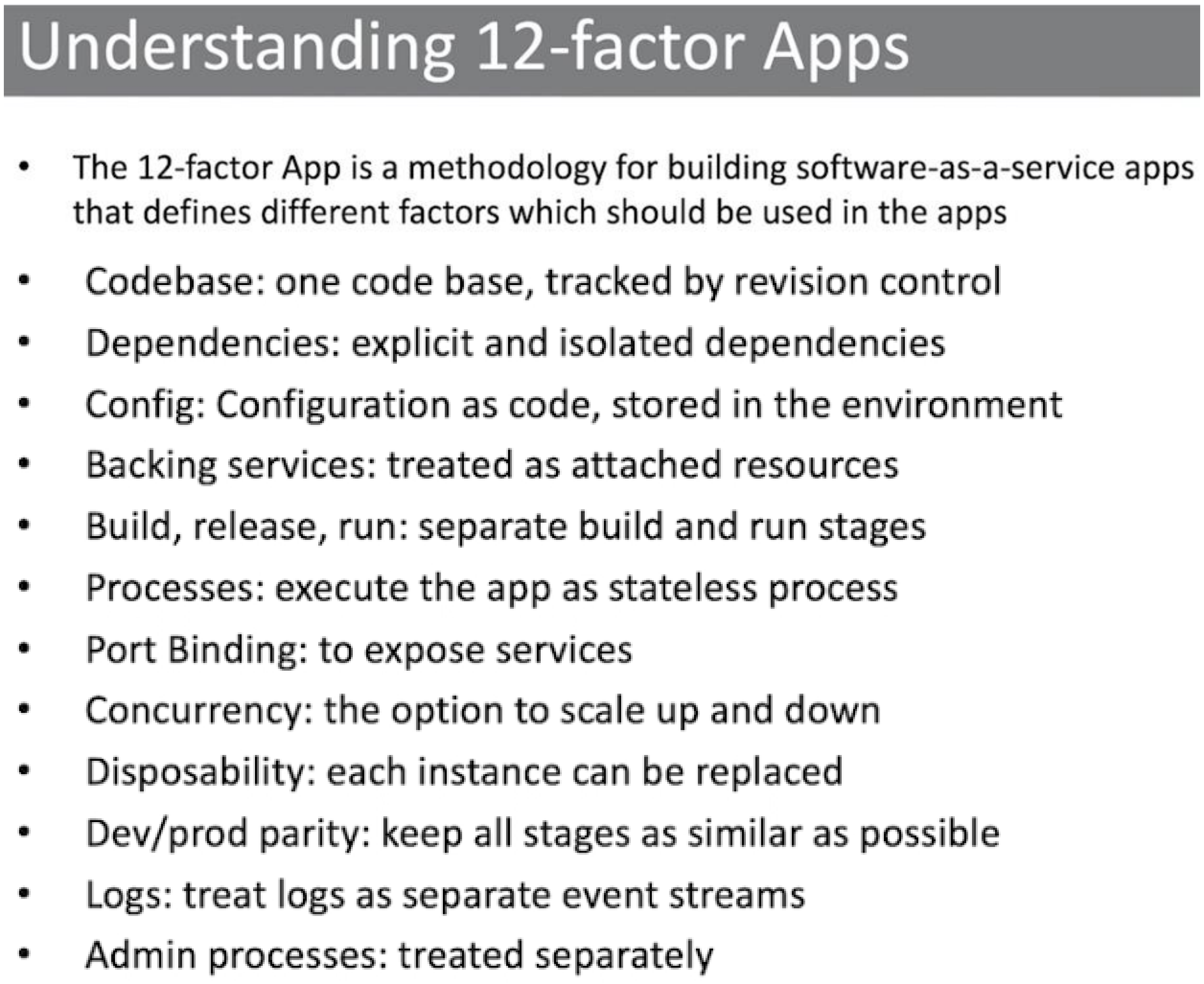
****



Understanding CI/CD Pipelines

- CI is Continuous Integration
- CD is Continuous Development
- CI/CD can be automated in a pipeline

****

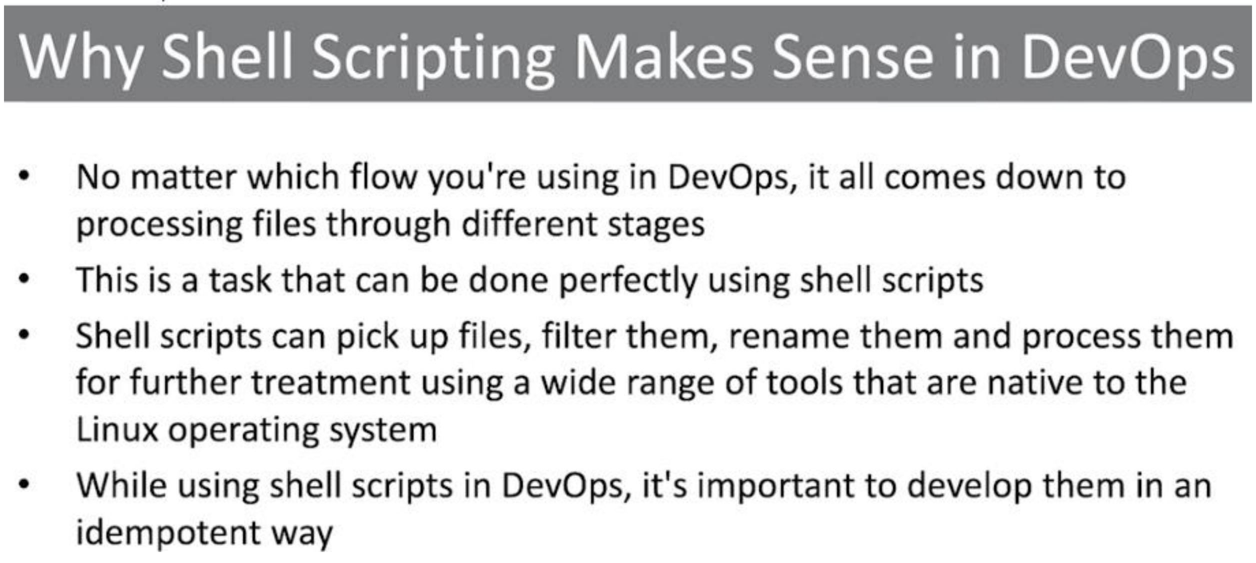


Understanding 12-factor Apps

- The 12-factor App is a methodology for building software-as-a-service apps that defines different factors which should be used in the apps
- Codebase: one code base, tracked by revision control
- Dependencies: explicit and isolated dependencies
- Config: Configuration as code, stored in the environment
- Backing services: treated as attached resources
- Build, release, run: separate build and run stages
- Processes: execute the app as stateless process
- Port Binding: to expose services
- Concurrency: the option to scale up and down
- Disposability: each instance can be replaced
- Dev/prod parity: keep all stages as similar as possible
- Logs: treat logs as separate event streams
- Admin processes: treated separately

****

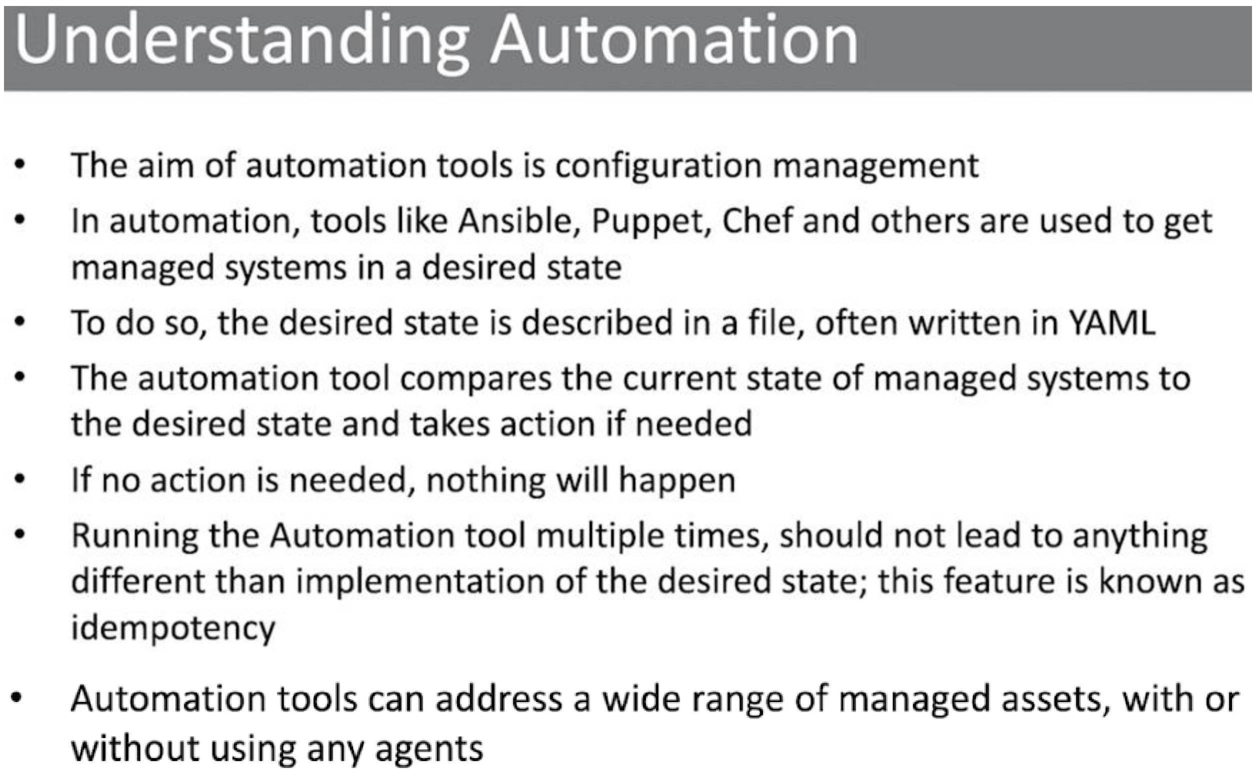
The Last One Means That: If We Run The Script Many Times, It Will Has The Same Output



Why Shell Scripting Makes Sense in DevOps

- No matter which flow you're using in DevOps, it all comes down to processing files through different stages
- This is a task that can be done perfectly using shell scripts
- Shell scripts can pick up files, filter them, rename them and process them for further treatment using a wide range of tools that are native to the Linux operating system
- While using shell scripts in DevOps, it's important to develop them in an idempotent way

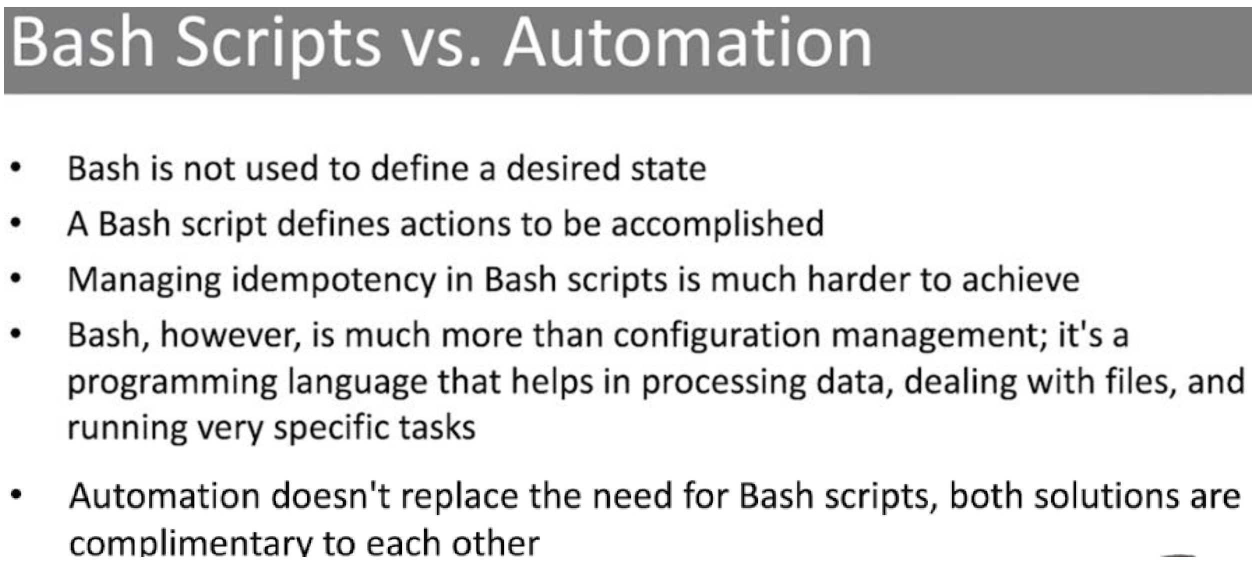
****



Understanding Automation

- The aim of automation tools is configuration management
- In automation, tools like Ansible, Puppet, Chef and others are used to get managed systems in a desired state
- To do so, the desired state is described in a file, often written in YAML
- The automation tool compares the current state of managed systems to the desired state and takes action if needed
- If no action is needed, nothing will happen
- Running the Automation tool multiple times, should not lead to anything different than implementation of the desired state; this feature is known as idempotency
- Automation tools can address a wide range of managed assets, with or without using any agents

****



Bash Scripts vs. Automation

- Bash is not used to define a desired state
- A Bash script defines actions to be accomplished
- Managing idempotency in Bash scripts is much harder to achieve
- Bash, however, is much more than configuration management; it's a programming language that helps in processing data, dealing with files, and running very specific tasks
- Automation doesn't replace the need for Bash scripts, both solutions are complimentary to each other

****