

# Bootcamp Devops Engineer

Encuentro #4

**Antes de empezar... journalctl!**

# Analizar logs del sistema en SystemD

- Mostrar los logs desde el inicio del sistema: `journalctl`
- Mostrar los logs para un servicio específico: `journalctl -u [servicio]`
- Mostrar los logs desde un tiempo específico: `journalctl --since "YYYY-MM-DD HH:MM:SS"`
- Mostrar los logs desde un la última hora: `journalctl --since "1 hour ago"`
- Mostrar los logs en follow mode: `journalctl -f`
- Mostrar los logs del sistema con detalles adicionales sobre errores o warnings:  
`journalctl -xe`

**Ahora sí... quiz!**

# Que vamos a ver hoy?

- Vim
- Bash Scripting
- Cron





# Vim

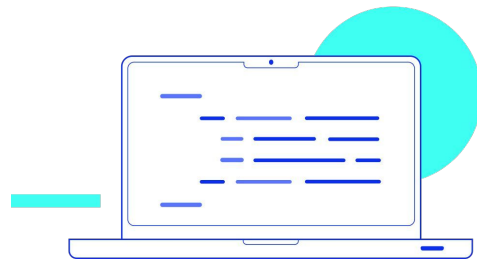


# Introducción a Vim

Vim es un editor de texto modal, altamente configurable, usado comúnmente en entornos Unix/Linux.

## Fortalezas:

- Eficacia y rapidez
- Configurabilidad
- Portabilidad
- Ligereza
- Extensibilidad
- Comunidad Activa



# Modos de Vim

## Modos Principales:

- **Normal:** Navegar y ejecutar comandos
- **Insert:** Editar texto
- **Visual:** Seleccionar texto
- **Command:** Ejecutar comandos específicos

## Cambio de Modos:

- **'Esc':** Salir a Normal
- **'i':** Para Insert
- **'v':** Para Visual
- **':'**: Para Command



# Navegación Básica

## Movimiento:

- **'h'**: Izquierda
- **'j'**: Abajo
- **'k'**: Arriba
- **'l'**: Derecha

## Búsqueda:

- **'/texto'**: Buscar hacia adelante
- **'?texto'**: Buscar hacia atrás

# Edición de Texto

## Comandos de Insert:

- **'i'**: Insertar antes del cursor
- **'a'**: Insertar después del cursor
- **'o'**: Nueva línea debajo
- **'O'**: Nueva línea arriba

## Borrar texto:

- **'x'**: Buscar carácter bajo el cursor
- **'dw'**: Buscar palabra
- **'dd'**: Borrar línea

# Comandos de Deshacer y Rehacer

## Deshacer:

- **'u':** Normal
- **':undo':** Comando

## Rehacer:

- **'Ctrl + r':** Normal
- **':redo':** Comando

# Uso del Modo Visual

## Selección de Texto:

- **'v'**: Visual
- **'V'**: Visual línea
- **'Ctrl' + 'v'**: Visual bloque

## Acciones en Texto Seleccionado:

- **'d'**: Borrar
- **'y'**: Copiar
- **'p'**: Pegar

# Comandos Comunes en Modo Command

## Guardar y Salir:

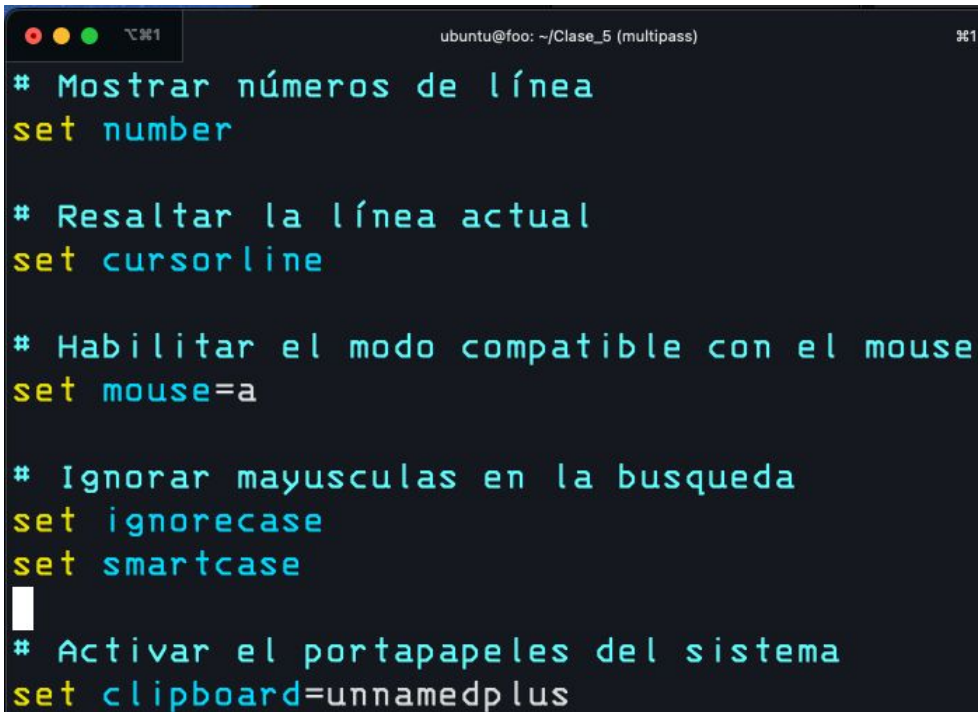
- `':w'`: Guardar
- `':q'`: Salir
- `':wq o :x'`: Guardar y salir
- `':q!'`: Salir sin guardar

## Comandos Variados:

- `':e nombre_archivo'`: Abrir un archivo
- `':s/viejo/nuevo/g'`: Reemplazar en la línea actual
- `':%s/viejo/nuevo/g'`: Reemplazar en todo el archivo
- `':!comando'`: Ejecutar comando en nuestra shell desde Vim
- `':set number'`: Mostrar números de línea

# Personalización Básica del .vimrc

El archivo .vimrc se utiliza para personalizar Vim según tus preferencias



```
ubuntu@foo: ~/Clase_5 (multipass) 961
# Mostrar números de línea
set number

# Resaltar la línea actual
set cursorline

# Habilitar el modo compatible con el mouse
set mouse=a

# Ignorar mayúsculas en la búsqueda
set ignorecase
set smartcase

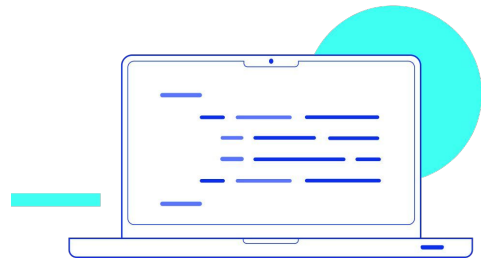
# Activar el portapapeles del sistema
set clipboard=unnamedplus
```

# Bash scripting

# Introducción a Bash

Bash (Bourne Again Shell) es un intérprete de comandos para Unix/Linux. Permite ejecutar comandos, automatizar tareas y escribir scripts.

Sus usos comunes suelen ser la automatización de tareas repetitivas, gestión de sistemas y el procesamiento de archivos.





# Estructura básica de un script

## Shebang:

- **#!/bin/bash**: Indica al sistema que interprete usar. En este caso, bash.

## Comentarios:

- **#**: Usamos el carácter '#' para agregar comentarios a nuestro código

# Variables

## Declaración de Variables:

- '[NOMBRE]=[VALOR]': Ejemplo:  
num1=10

## Uso de Variables:

- 'echo "El primer número es igual a \$num1"'

# Input del Usuario

## Leer Input del Usuario

- **'read'**: Usamos la keyword 'read' para leer input del usuario. Ejemplo:

```
echo "Introduce tu nombre:"  
read NOMBRE  
echo "Hola, $NOMBRE"  
~
```

# Estructuras de Control

Condicionales:

```
if [ $NOMBRE == "Juan" ]; then
    echo "Hola, Juan"
else
    echo "No sos Juan"
fi
```

Bucle 'for':

```
for i in 1 2 3 4 5; do
    echo "Número $i"
done
```

# Estructuras de Control

Bucle 'while':

```
COUNTER=1
while [ $COUNTER -le 5 ]; do
    echo "Contador: $COUNTER"
    ((COUNTER++))
done
```

# Funciones

```
funcion_saludo() {  
    echo "Hola, $1"  
}  
funcion_saludo "Juan"
```

## Iniciar nuestra instancia Linux

- Crear la instancia - `multipass launch -n <nombre>`
- Tomar la shell de nuestra instancia - `multipass shell <nombre>`

## Clonar repositorio de la clase

- `git clone https://github.com/JCaimo/Clase5.git`

## Ejercicios =)

- Mostrar en pantalla 'Hola, Mundo!'
- Solicitar al usuario ingresar dos números y sumarlos
- Solicitar al usuario un numero, y verificar si es par o impar
- Imprimir los números del 1 al 10 con 'for'
- Solicitar al usuario ingresar números hasta que el número ingresado sea 0. Una vez ingrese 0, sumar todos los números ingresados
- Crear un archivo llamado 'alumnos.txt' que agregue una lista de nombres. Luego, leer e imprimir el contenido del archivo
- Definir una función que imprima 3 veces en pantalla 'Hola, Mundo!'
- Aceptar dos argumentos, y al ejecutarse imprimir la suma
- Encontrar y contar todas las líneas un archivo de texto que contengan una palabra específica



## Ejercicios (más difíciles) =)

- Escribir un script que haga un backup de archivos y los guarde en un directorio de respaldo con la fecha actual
- Escribir un script que archive y comprima los archivos de logs que superen un tamaño determinado
- Escribir un script que sincronice archivos de un directorio local a un disco montado utilizando rsync
- Escribir un script que cree un nuevo usuario, le asigne un grupo específico y configure permisos para un directorio.

# Cron

# Introducción a Cron

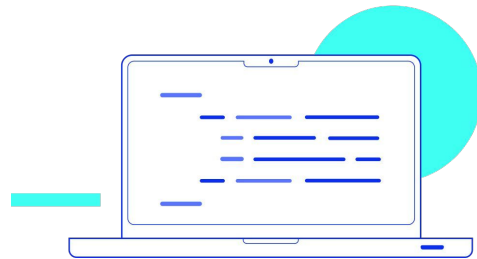
Cron es un servicio de Unix/Linux que se utiliza para programar la ejecución automática de tareas en intervalos específicos.

## Terminología:

- **Cron Job:** Tarea programada en cron
- **Crontab:** Archivo que contiene la programación de los cron jobs

## Usos comunes:

- Realizar copias de seguridad
- Ejecutar scripts de mantenimiento



# Sintaxis de crontab

## Formato básico:

- `* * * * *` comando\_a\_ejecutar

## Campos:

- Minuto (0 - 59)
- Hora (0 - 23)
- Día del mes (1 - 31)
- Mes (1 - 12)
- Día de la semana (0 - 7)

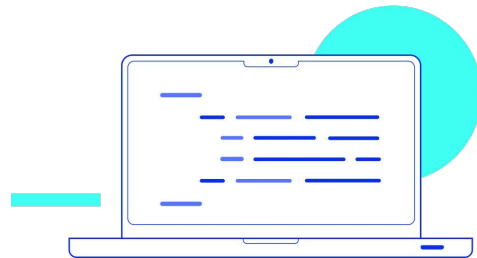
## Ejemplo:

- Ejecutar un script todos los días a las 12 de la noche: `0 0 * * * /home/user/script.sh`



## Usos prácticos de crontab

- Editar el crontab del usuario: `crontab -e`
- Listar los cron jobs actuales: `crontab -l`
- Eliminar todos los cron jobs: `crontab -r`



# Recomendaciones

- VIM:
  - Inglés
    - @ThePrimeagen
  - Español
    - @PeladoNerd
  - Investigar NeoVim, y NVChad

