

# Introducción a Bash Scripting

Clase 3 – 7/3/2024



Bootcamp de DevOps @ Código Facilito



# Alejandra Bricio



cloud, scripting, k8s  
@alebricio

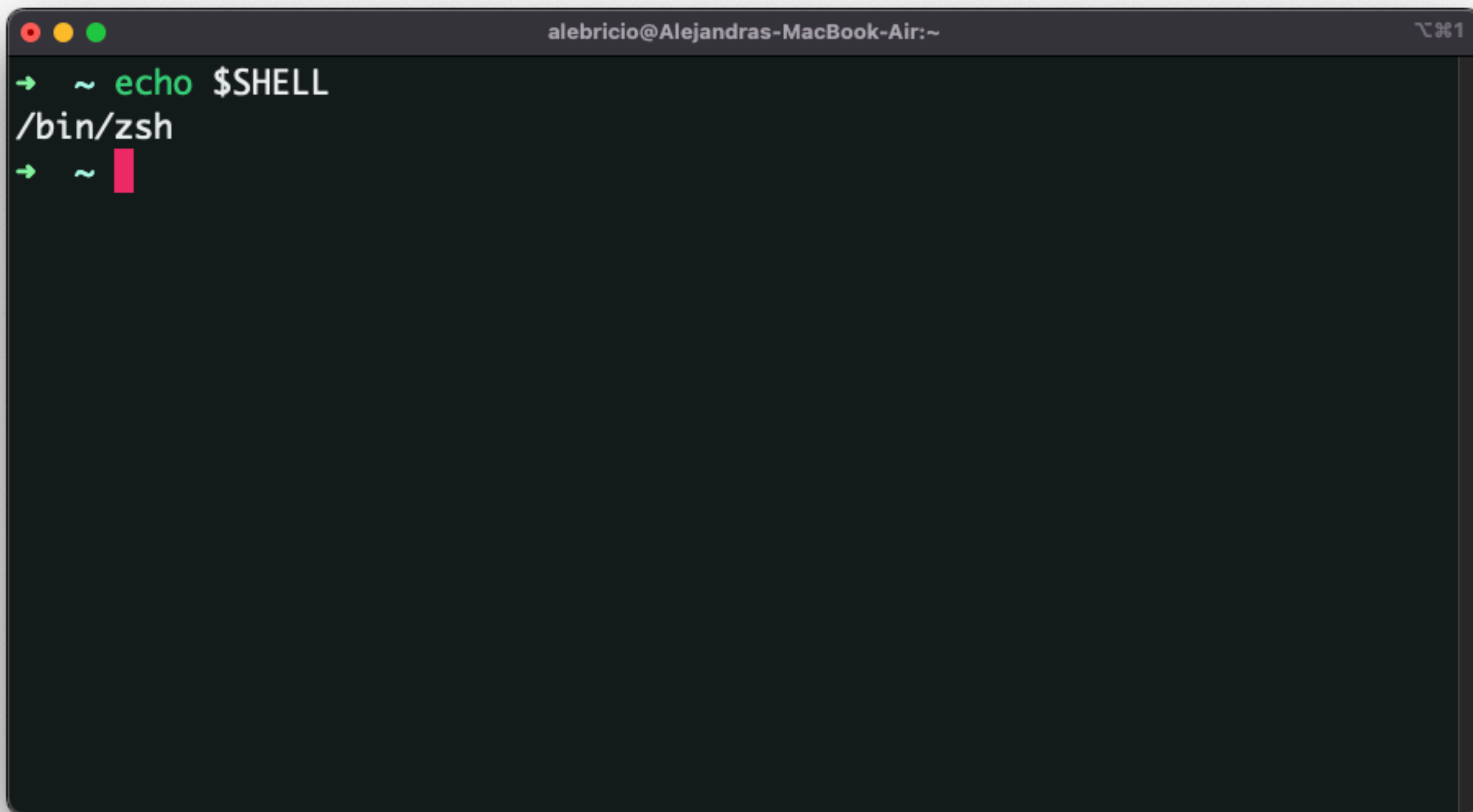
- Site Reliability Engineer @ PayPal
- AWS Solutions Architect – Associate
- AWS Jalisco User Group – Leader
- AWS Community Day Mexico – Organizadora
- Tutora Bootcamp de Ciencias Computacionales @ Código Facilito
  - Redes Computacionales
  - Sistemas Distribuidos

# Introducción a Bash Scripting – Temario

- Introducción
- Bash
  - Shell, Terminales
  - Usando bash
    - Navegando bash
    - Otras utilidades (comandos)
- Bash Scripting
  - Sintaxis básica de bash
  - Variables, operadores y estructuras de control

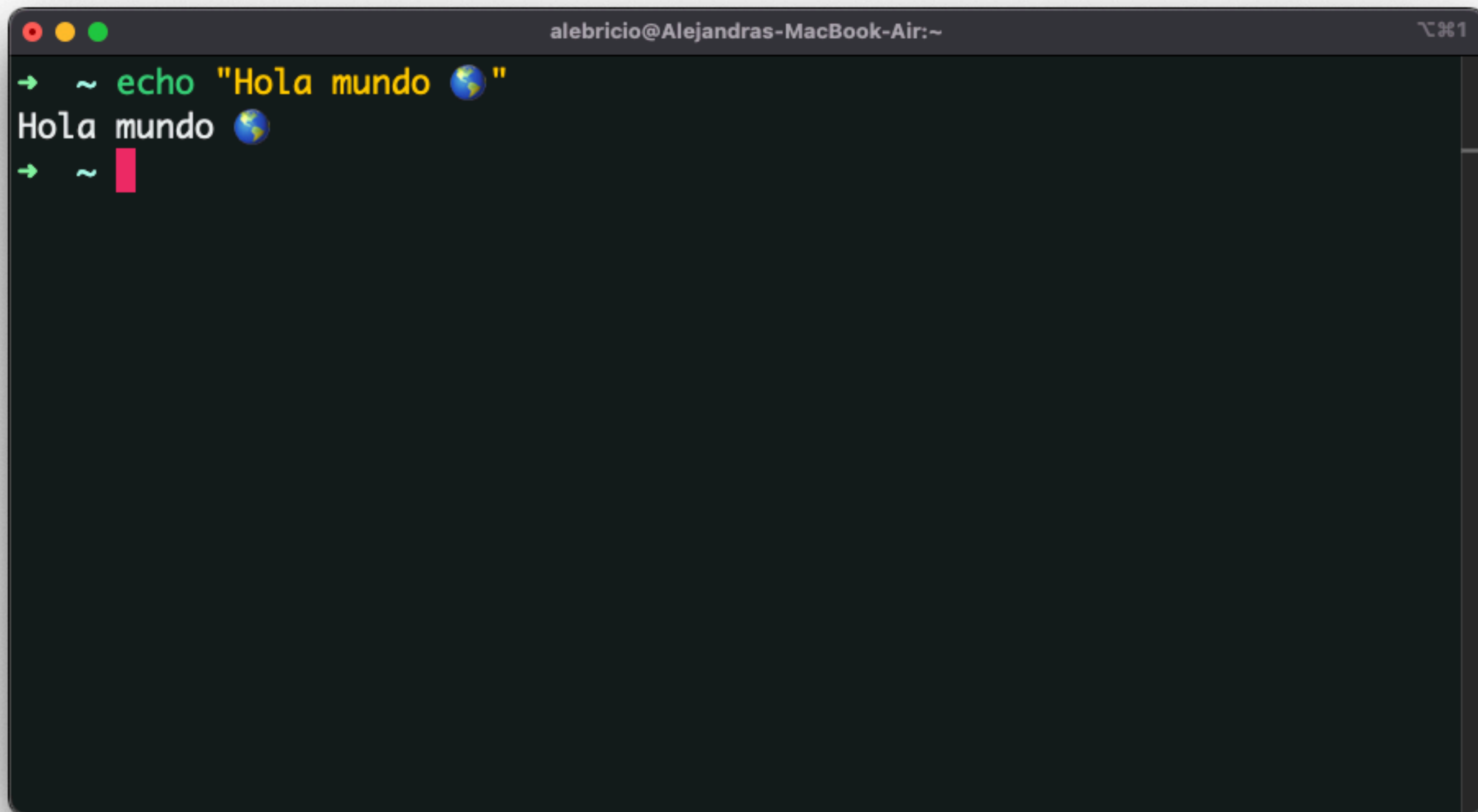
# Antes de empezar...

- Para este curso, necesitas estar usando un shell Unix como Bash o ZSH. Si estás en Linux o macOS, no tienes que hacer nada especial. Si estás en Windows, necesitas asegurarte de que no estás ejecutando cmd.exe o PowerShell; puedes usar **Windows Subsystem para Linux** o una máquina virtual Linux para usar herramientas de línea de comandos estilo Unix.
- Para asegurarte de que estás ejecutando un shell apropiado, puedes probar el comando `echo $SHELL`. Si dice algo como `/bin/bash` o `/usr/bin/zsh`, significa que estás ejecutando el programa adecuado.



```
alebricio@Alejandras-MacBook-Air:~  
→ ~ echo $SHELL  
/bin/zsh  
→ ~
```

A terminal window on a MacBook Air. The title bar shows the user 'alebricio' and the machine name 'Alejandras-MacBook-Air'. The prompt is '~'. The user has entered the command 'echo \$SHELL' and the output is '/bin/zsh'. The prompt is now '~' with a red cursor.



```
→ ~ echo "Hola mundo 🌐"  
Hola mundo 🌐  
→ ~
```

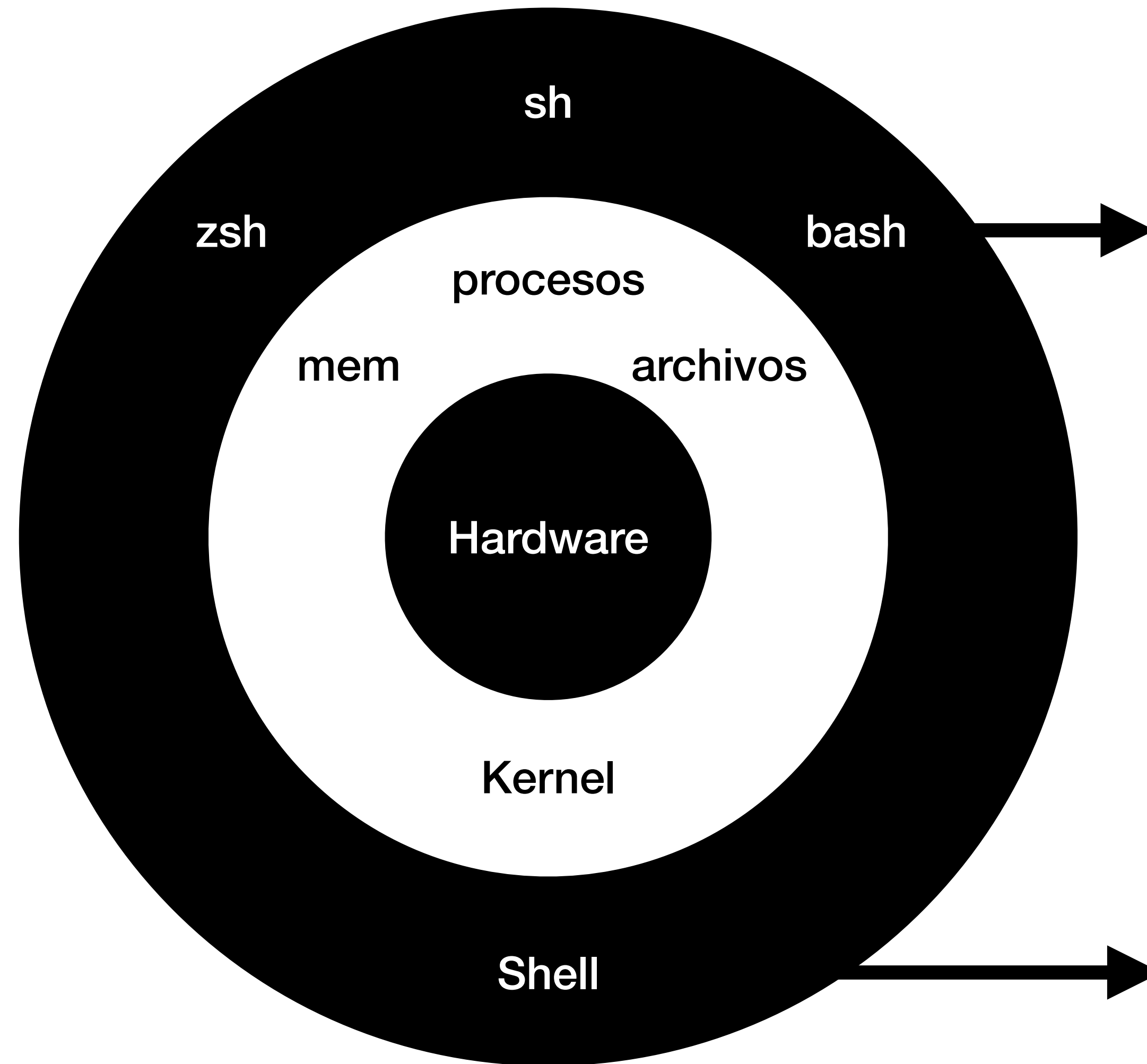
The image shows a macOS-style terminal window with a dark background. The title bar at the top contains three colored window control buttons (red, yellow, green) on the left, the text 'alebricio@Alejandras-MacBook-Air:~' in the center, and a zoom icon and the number '1' on the right. The terminal content shows a command prompt '→' followed by '~' and the command 'echo "Hola mundo 🌐"'. The output of the command is 'Hola mundo 🌐'. Below this, another prompt '→' followed by '~' and a red cursor block is visible.

# ¿Qué es un **script**?

Los **scripts** ayudan a escribir una secuencia de comandos en un archivo y luego ejecutarlos.

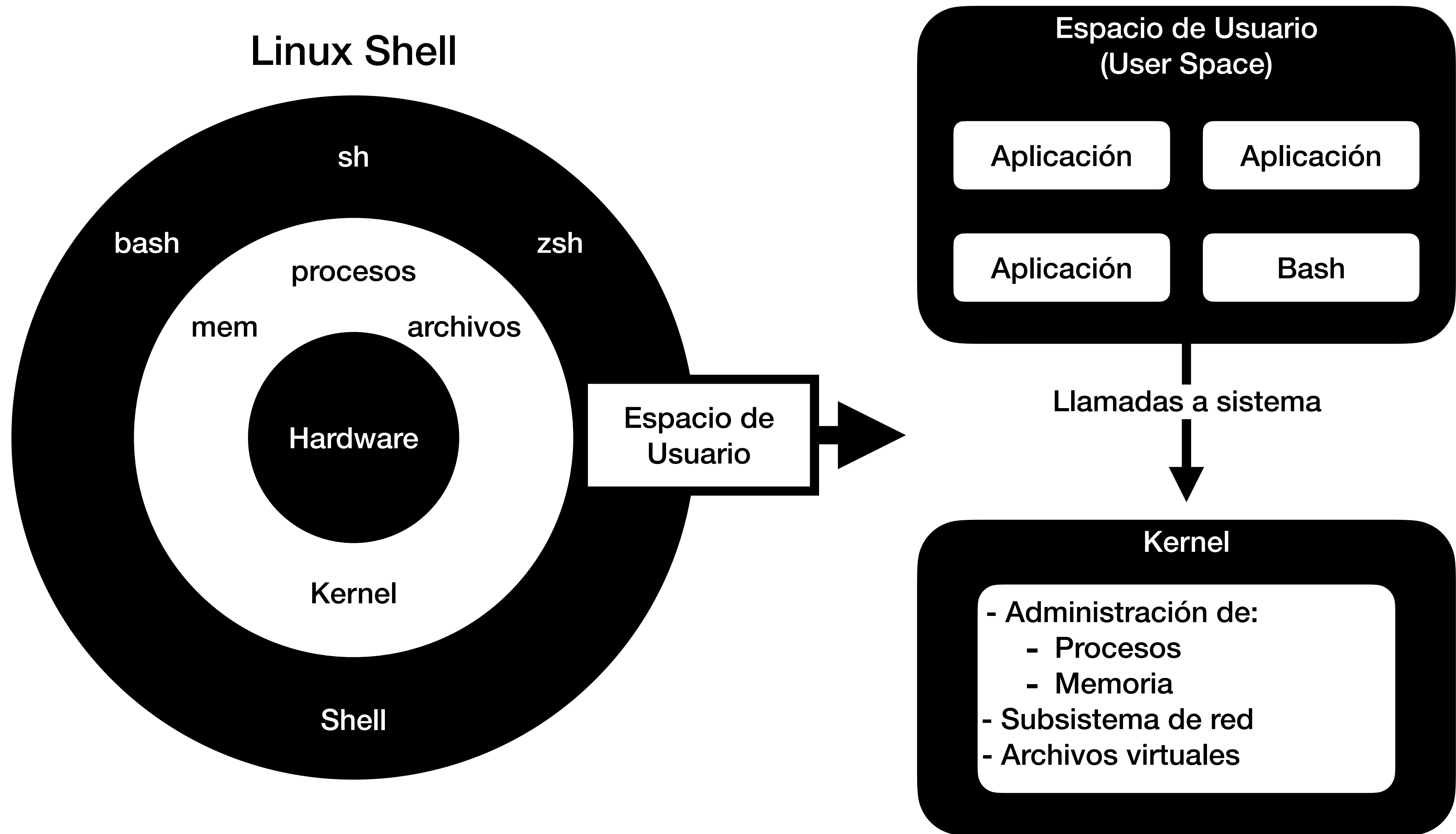
¿Qué es **bash**?

## Linux Shell



Es un tipo  
de shell.

Es un programa que  
expone los servicios  
de un sistema  
operativo a un usuario  
o a otros programas.



# Demo

(bash, dtruss, cat, python)

**Shell**

```
graph TD; Shell[Shell] --> bash[bash]; Shell --> zsh[zsh]; Shell --> sh[sh];
```

**bash**

**zsh**

**sh**

# Bash

- Mas usuarios y soporte
- Mas compatibilidad con sistemas Linux

# Zsh

- Mas configurable
  - OhMyZsh
- Tab para navegar entre opciones
- Autocompletado es más rapido
- Syntax highlighting

¿Qué es la **terminal**?

# Terminales

```
graph TD; A[Terminales] --> B[Terminal]; A --> C[iTerm2]
```

**Terminal**

**iTerm2**



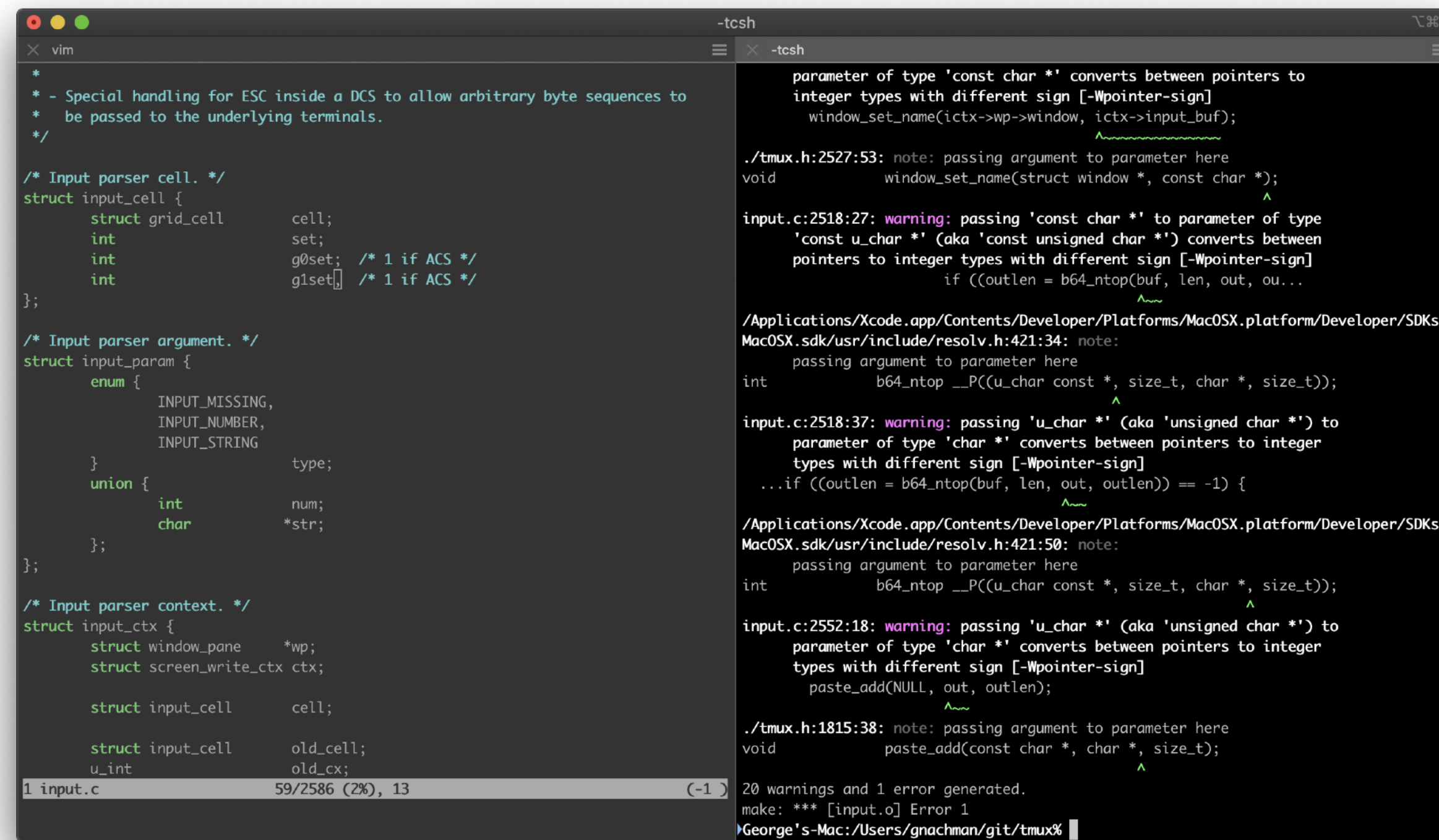
# iTerm2

iTerm2 is a terminal emulator for macOS that does amazing things.

<https://iterm2.com/features.html>

# iTerm2 – Paneles

Divida una pestaña en varios paneles, cada uno de los cuales mostrará una sesión diferente. Puede cortar vertical y horizontalmente y crear cualquier número de paneles en cualquier disposición imaginable.



```
-tosh
vim
/*
 * - Special handling for ESC inside a DCS to allow arbitrary byte sequences to
 *   be passed to the underlying terminals.
 */

/* Input parser cell. */
struct input_cell {
    struct grid_cell    cell;
    int                 set;
    int                 g0set; /* 1 if ACS */
    int                 g1set; /* 1 if ACS */
};

/* Input parser argument. */
struct input_param {
    enum {
        INPUT_MISSING,
        INPUT_NUMBER,
        INPUT_STRING
    } type;
    union {
        int num;
        char *str;
    };
};

/* Input parser context. */
struct input_ctx {
    struct window-pane *wp;
    struct screen_write_ctx ctx;

    struct input_cell cell;

    struct input_cell old_cell;
    u_int              old_cx;
};

1 input.c 59/2586 (2%), 13 (-1)

parameter of type 'const char *' converts between pointers to
integer types with different sign [-Wpointer-sign]
window_set_name(ictx->wp->window, ictx->input_buf);

./tmux.h:2527:53: note: passing argument to parameter here
void window_set_name(struct window *, const char *);

input.c:2518:27: warning: passing 'const char *' to parameter of type
'const u_char *' (aka 'const unsigned char *') converts between
pointers to integer types with different sign [-Wpointer-sign]
if ((outlen = b64_ntop(buf, len, out, ou...

/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/
MacOSX.sdk/usr/include/resolv.h:421:34: note:
passing argument to parameter here
int b64_ntop __P((u_char const *, size_t, char *, size_t));

input.c:2518:37: warning: passing 'u_char *' (aka 'unsigned char *') to
parameter of type 'char *' converts between pointers to integer
types with different sign [-Wpointer-sign]
...if ((outlen = b64_ntop(buf, len, out, outlen)) == -1) {

/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/
MacOSX.sdk/usr/include/resolv.h:421:50: note:
passing argument to parameter here
int b64_ntop __P((u_char const *, size_t, char *, size_t));

input.c:2552:18: warning: passing 'u_char *' (aka 'unsigned char *') to
parameter of type 'char *' converts between pointers to integer
types with different sign [-Wpointer-sign]
paste_add(NULL, out, outlen);

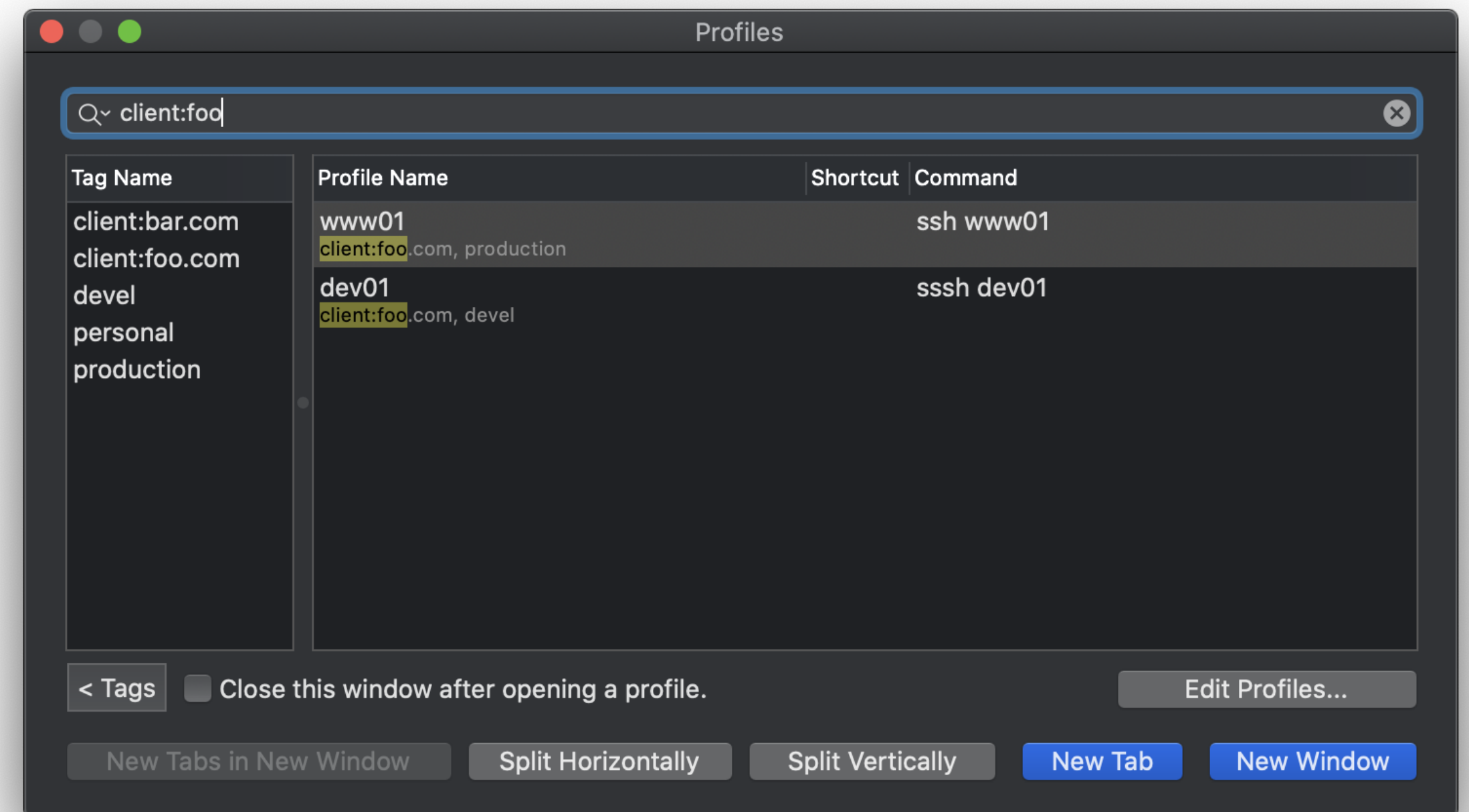
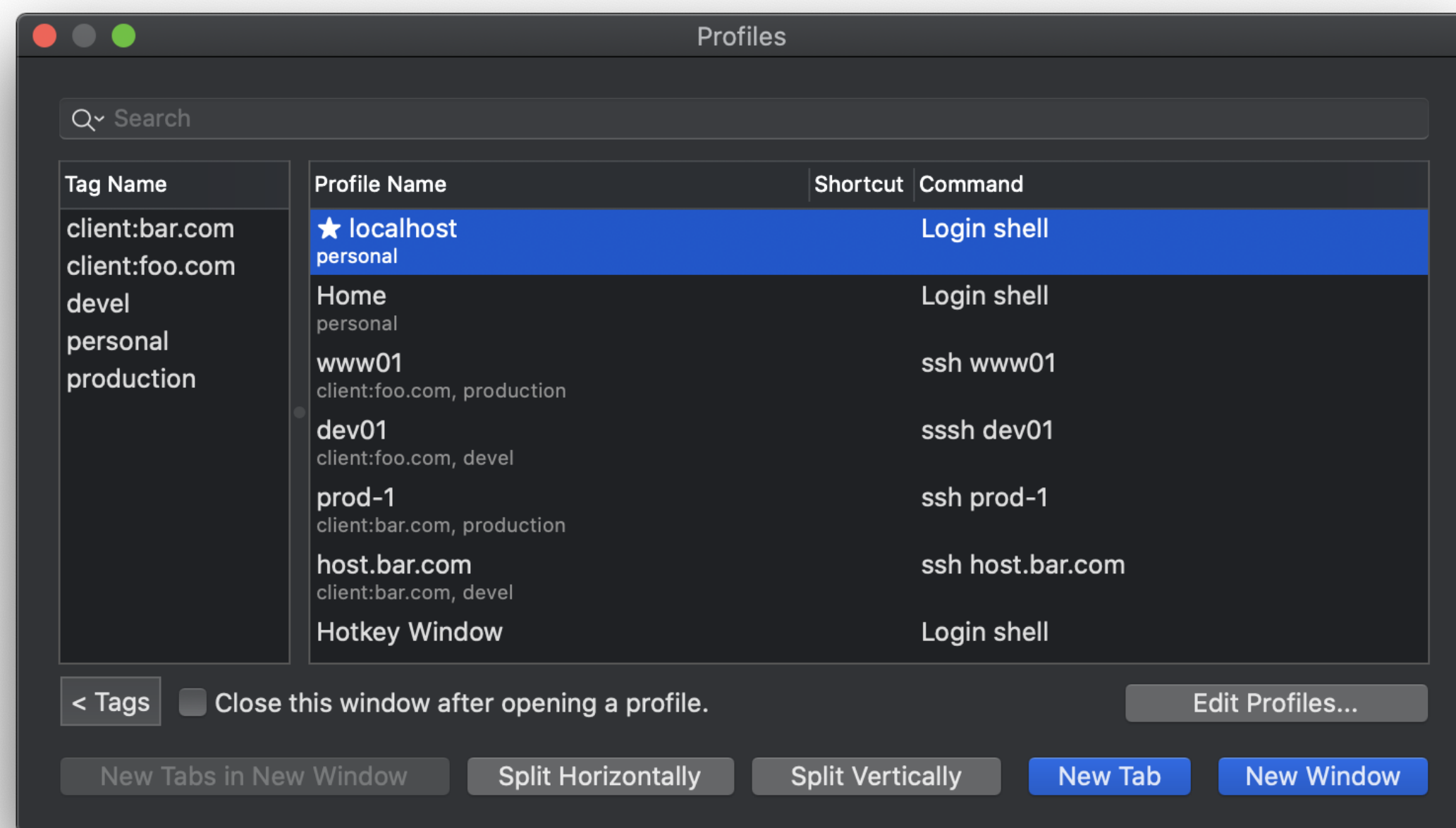
./tmux.h:1815:38: note: passing argument to parameter here
void paste_add(const char *, char *, size_t);

20 warnings and 1 error generated.
make: *** [input.o] Error 1
George's-Mac:/Users/gnachman/git/tmux% 
```

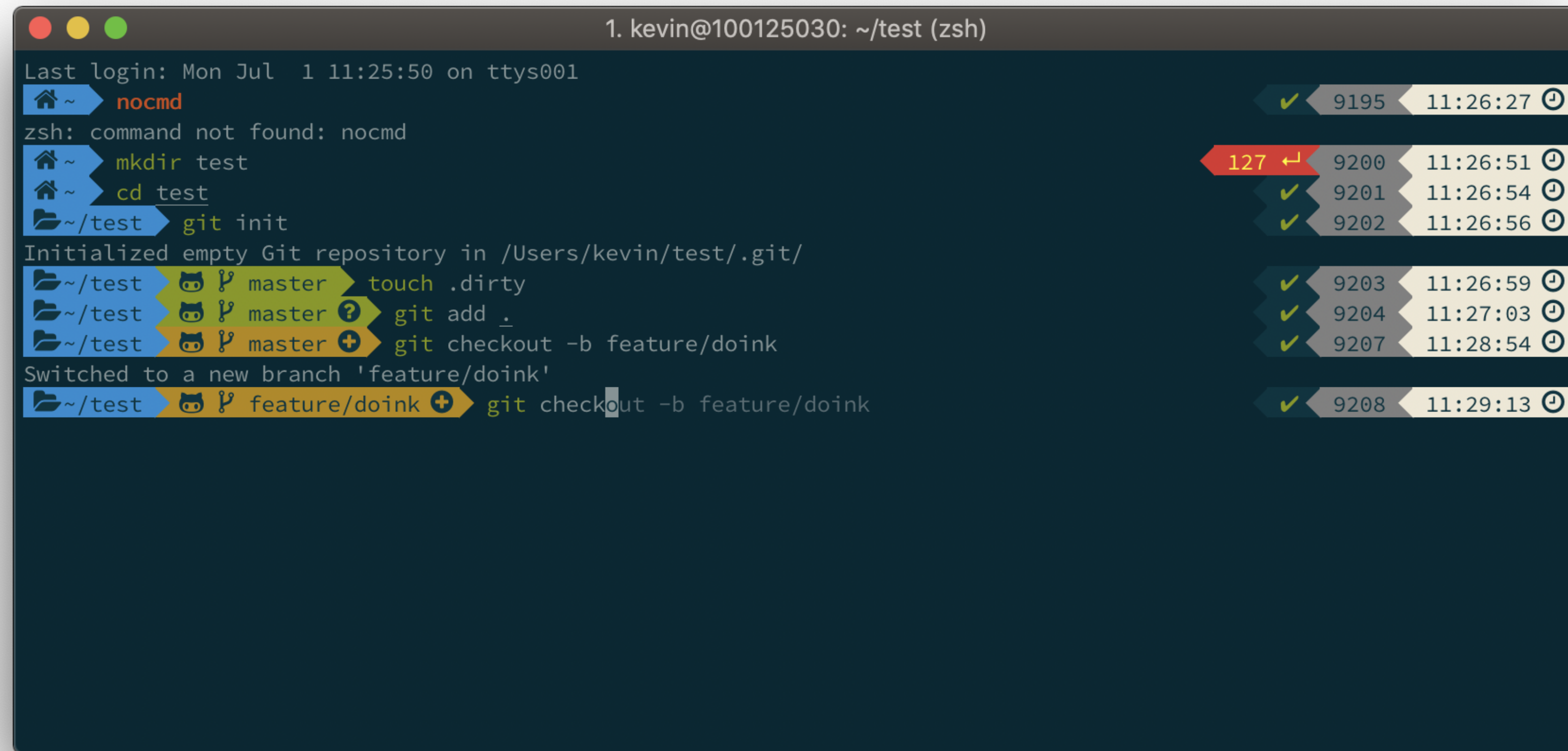
<https://iterm2.com/features.html>

# iTerm2 – Profiles

¿Necesita almacenar configuraciones separadas para muchos hosts diferentes? iTerm2 proporciona una base de datos de perfiles etiquetable y con capacidad de búsqueda para que pueda encontrar fácilmente el perfil que está buscando.



# Tutorial configuración iTerm2



The screenshot shows an iTerm2 terminal window with a dark blue background. The title bar at the top reads "1. kevin@100125030: ~/test (zsh)". The terminal content shows a series of commands and their outputs, with a status bar on the right indicating the execution status of each command.

```
Last login: Mon Jul 1 11:25:50 on ttys001
~ ➤ nocmd
zsh: command not found: nocmd
~ ➤ mkdir test
~ ➤ cd test
~/test ➤ git init
Initialized empty Git repository in /Users/kevin/test/.git/
~/test ➤ touch .dirty
~/test ➤ git add .
~/test ➤ git checkout -b feature/doink
Switched to a new branch 'feature/doink'
~/test ➤ git checkout -b feature/doink
```


The status bar on the right shows the following information for each command:

Command	Status	Line Number	Time
nocmd	✓	9195	11:26:27
mkdir test	127 ↵	9200	11:26:51
cd test	✓	9201	11:26:54
git init	✓	9202	11:26:56
touch .dirty	✓	9203	11:26:59
git add .	✓	9204	11:27:03
git checkout -b feature/doink	✓	9207	11:28:54
git checkout -b feature/doink	✓	9208	11:29:13

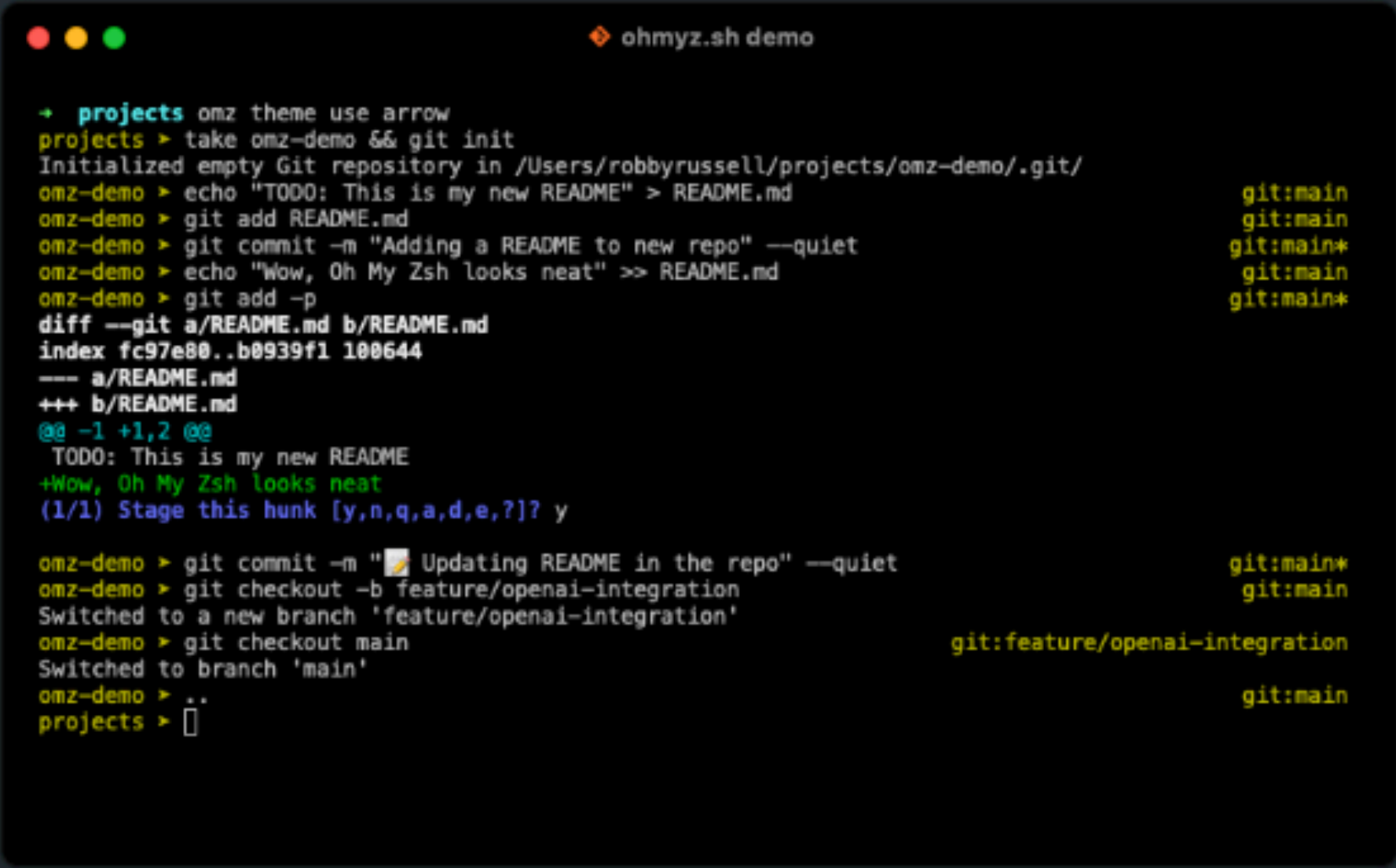
[https://www.youtube.com/watch?v=D2PSsnqgBiw&t=990s&ab\\_channel=QuentinWattTutorials](https://www.youtube.com/watch?v=D2PSsnqgBiw&t=990s&ab_channel=QuentinWattTutorials)

<https://gist.github.com/kevin-smets/8568070>

# oh my zsh



DOCSCODETHEMESPLUGINS SHOPSPONSOR



```
+ projects omz theme use arrow
projects > take omz-demo && git init
Initialized empty Git repository in /Users/robbyrussell/projects/omz-demo/.git/
omz-demo > echo "TODO: This is my new README" > README.md
omz-demo > git add README.md
omz-demo > git commit -m "Adding a README to new repo" --quiet
omz-demo > echo "Wow, Oh My Zsh looks neat" >> README.md
omz-demo > git add -p
diff --git a/README.md b/README.md
index fc97e80..b0939f1 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
  TODO: This is my new README
+Wow, Oh My Zsh looks neat
(1/1) Stage this hunk [y,n,q,a,d,e,?]? y

omz-demo > git commit -m "📄 Updating README in the repo" --quiet
omz-demo > git checkout -b feature/openai-integration
Switched to a new branch 'feature/openai-integration'
omz-demo > git checkout main
Switched to branch 'main'
omz-demo > ..
projects > █
```

git:main  
git:main  
git:main\*  
git:main  
git:main\*

git:main\*  
git:main

git:feature/openai-integration


git:main

## *Unleash* your terminal like never before.

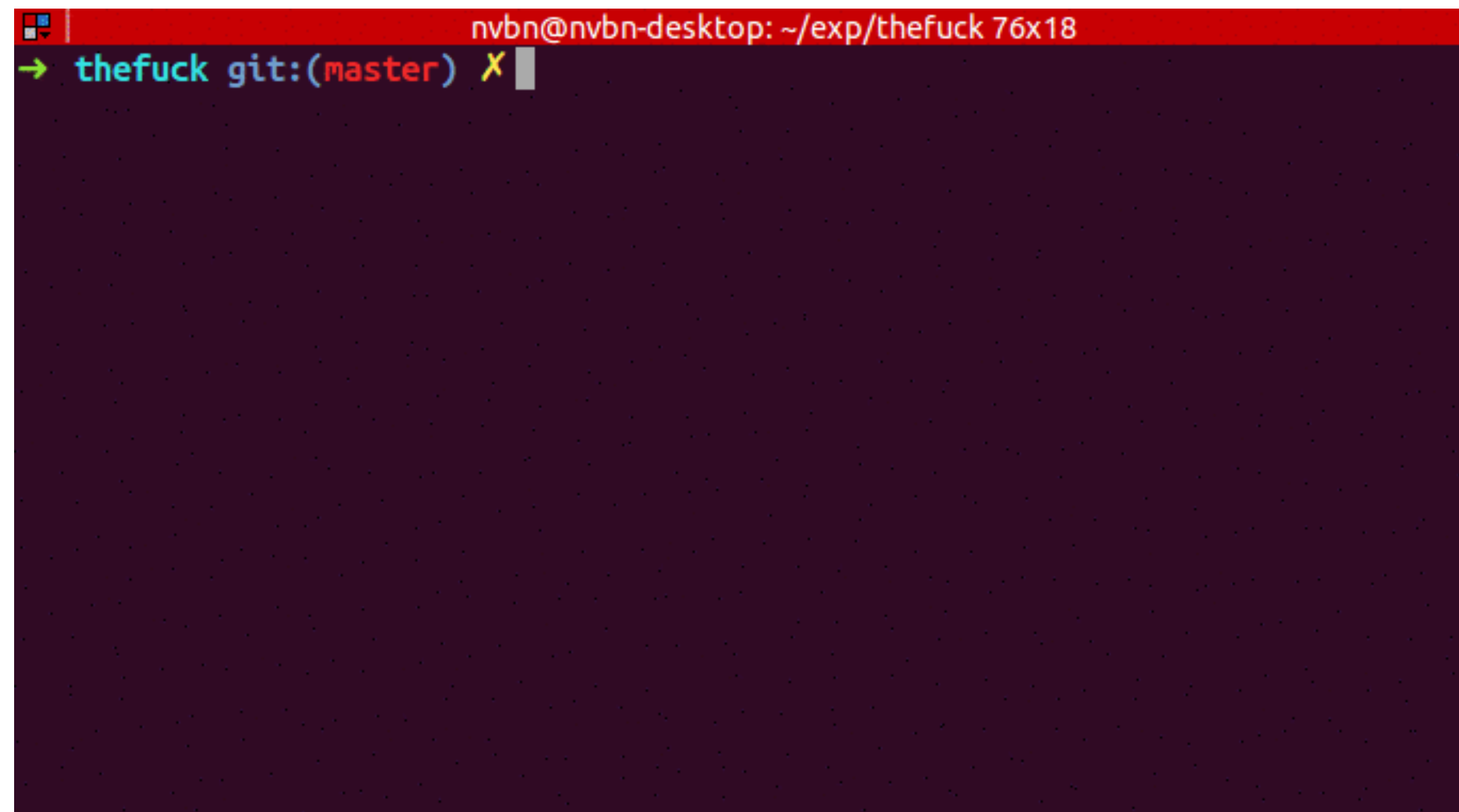
Oh My Zsh is a delightful, open source, community-driven framework for managing your Zsh configuration. It comes bundled with thousands of helpful functions, helpers, plugins, themes, and a few things that make you shout...

### "Oh My ZSH!"

Install oh-my-zsh

 batteries included.

# oh my zsh – Plugins



A terminal window with a red title bar showing the user 'nvbn' on a desktop. The prompt is 'nvbn@nvbn-desktop: ~/exp/thefuck 76x18'. The user has entered 'thefuck git:(master)' and a yellow 'X' icon is visible next to the prompt, indicating the 'thefuck' plugin is active.

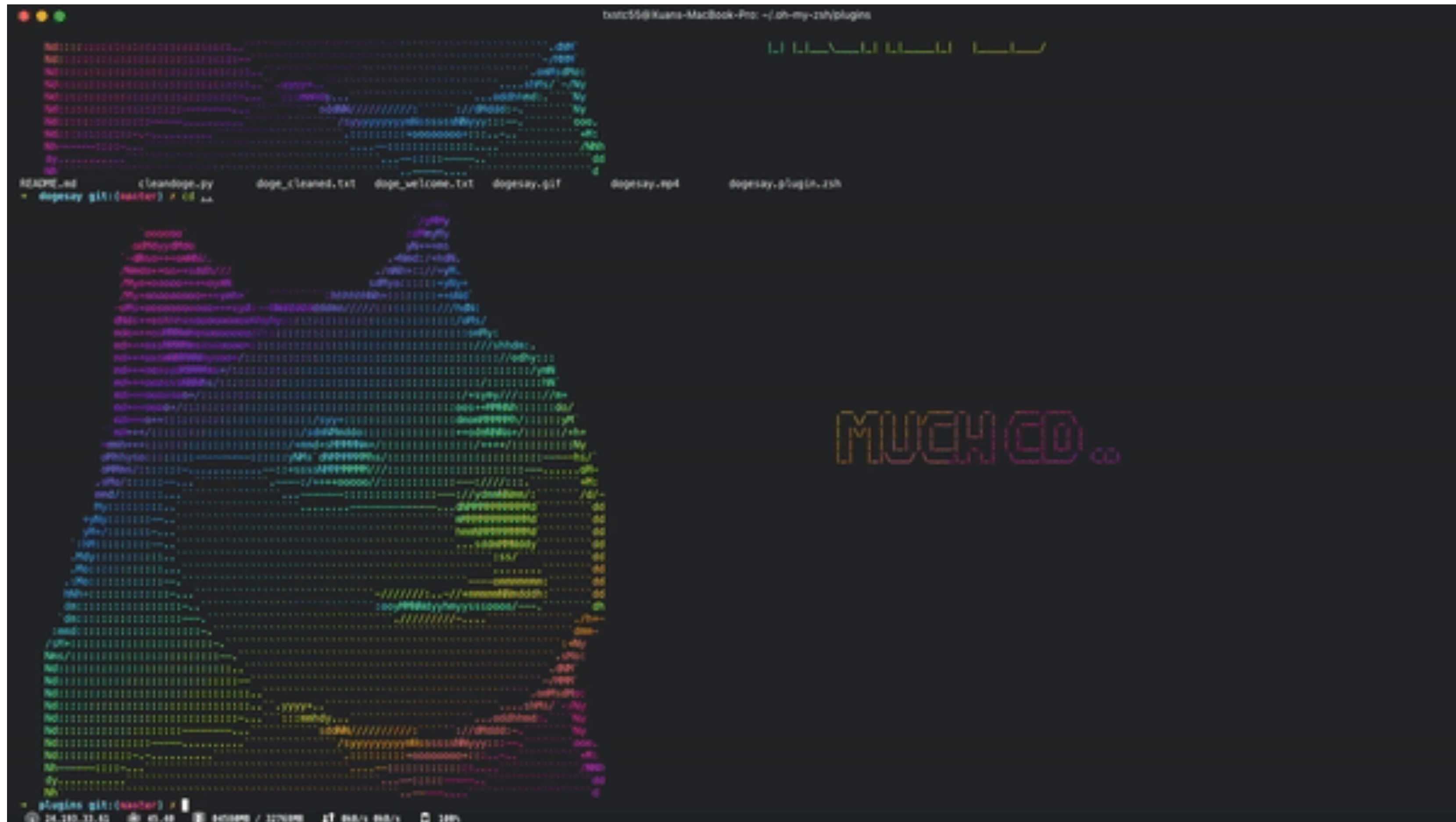
Before: % echo \$'Hello, world\x21 '  
After: % echo \$'Hello, world\x21 '

Before: % echo 'Pay \$5 to Joe'  
After: % echo 'Pay \$5 to Joe'

Before: % ( foo=42 }  
After: % ( foo=42 }

Before: % echo 3> /proc/.../vm/drop\_caches  
After: % echo 3> /proc/.../vm/drop\_caches

# oh my zsh – Plugins



<https://github.com/ohmyzsh/ohmyzsh/wiki/Plugins-Overview>

# Inicialización del shell



# Archivos de inicialización

## Bash

- `.bash_profile`
- `/.bash_profile`
- ...

## Zsh

- `.zshenv`
- `.zprofile`
- `.zshrc`
- ...

# **Variables de entorno**

- **printenv**

# ¿Qué es el PATH?

- La variable de entorno global más importante que debes configurar.
- Esta es la variable que le dice al shell bash dónde encontrar diferentes archivos ejecutables y scripts.
- El shell comprobará los directorios listados en la variable PATH para el script que estás intentando encontrar.

# Comandos de Bash



**Cheatsheet (Hoja de trucos) de comandos**

[https://www.geeksforgeeks.org/  
linux-commands-cheat-sheet/](https://www.geeksforgeeks.org/linux-commands-cheat-sheet/)

# Gestión de archivos y directorios

- **ls:** lista los archivos y directorios.
- **cd:** cambia el directorio actual.
- **cp:** copia archivos o directorios.
- **mv:** mueve o renombra archivos o directorios.
- **rm:** elimina archivos o directorios.
- **mkdir:** crea nuevos directorios.

# Manipulación de texto

- **grep:** busca patrones en el texto.
- **sed:** editor de texto que realiza procesamiento basado en patrones.
- **awk:** lenguaje de programación diseñado para procesar y analizar archivos de texto.
- **cat:** concatena y muestra el contenido de archivos.
- **sort:** ordena las líneas de texto.
- **uniq:** reporta o filtra líneas repetidas en un archivo.

# Administración de procesos

- **ps:** muestra los procesos en ejecución.
- **top:** muestra una lista en tiempo real de los procesos en ejecución.
- **kill:** envía una señal a un proceso para terminarlo.

# Gestión del sistema

- **df:** muestra el uso del disco.
- **ifconfig/ip:** configura o muestra la configuración de red.
- **chmod:** cambia los permisos de acceso de archivos o directorios.
- **chown:** cambia el propietario de archivos o directorios.

# Red y comunicaciones

- **ping:** comprueba la conectividad con otro host.
- **wget:** descarga archivos desde Internet.
- **ssh:** inicia sesión en otro computador vía red.
- **scp:** copia archivos entre hosts en una red.
- **netstat:** muestra las conexiones de red, tablas de enrutamiento, estadísticas de interfaces, etc.

# Administración de paquetes

- **apt-get** (en sistemas basados en Debian): herramienta para manejar paquetes.
- **yum** (en sistemas basados en RedHat): gestor de paquetes.
- **dpkg**: herramienta de gestión de paquetes en Debian.
- **rpm**: gestor de paquetes para RedHat, CentOS, y similares.

**Cheatsheet (Hoja de trucos) de comandos**

[https://www.geeksforgeeks.org/  
linux-commands-cheat-sheet/](https://www.geeksforgeeks.org/linux-commands-cheat-sheet/)

# Pequeño break ~



# Mi primer script de Bash



# Lenguaje Bash

- Interpretado
- Extension de archivo: `.sh`
- Script inicia con: `#!/bin/bash`
- Permisos de ejecución
- Color de archivo

# Mi primer script de Bash

```
touch <miarchivo>  
chmod +x <miarchivo>  
vi <miarchivo>
```

# Definiendo variables



```
#!/bin/bash  
# Ejemplo de variable  
saludo=Hola  
nombre=Cody  
echo $saludo $nombre
```

# Variables especiales

- `$0` representa el nombre del script
- `$1 – $9` los primeros nueve argumentos que se pasan a un script en Bash
- `$@` todos los argumentos que se han pasado al script
- `$?` la salida del último proceso que se ha ejecutado

# Leyendo input de usuarios



```
#!/bin/bash
```

```
echo "Ingresa un numero"  
read a
```

```
echo "Ingresa un numero"  
read b
```

```
var=$((a+b))  
echo $var
```

# Operadores Lógicos y Sentencias condicionales



```
if [ condiciones ]  
    then  
        ejecutar comandos  
fi
```



```
if [[ condicion ]]  
then  
    sentencia  
elif [[ condicion ]]; then  
    sentencia  
else  
    do hacer esto por defecto  
fi
```



```
#!/bin/bash
```

```
read x  
read y
```

```
if [ $x -gt $y ]  
then  
    echo X es mayor que Y  
elif [ $x -lt $y ]  
then  
    echo X es menor que Y  
elif [ $x -eq $y ]  
then  
    echo X es igual que Y  
fi
```

# For Loops



```
#!/bin/bash
```


```
for i in {1..5}  
do  
    echo $i  
done
```



```
#!/bin/bash
```

```
for X in rojo verde azul  
do  
    echo $X  
done
```

# While Loops



```
#!/bin/bash

LINEA=1

while read -r LINEA
do
    echo "$LINEA: $LINEA_ACTUAL"
    ((LINEA++))
done < "archivo_ejemplo.txt"
```

# ¿Preguntas?



@alebricio

# Tarea



@alebricio

# ¡ Gracias !



@alebricio