4TH LABORATORIUM

Marcia L. Baptista
2024/2025

AGENDA

Introduction

Why UNIX Commands?

How to run Shell Commands?

Tip

Shell commands in Databricks

Operations

Distributed File System

INTRODUCTION

On this Lab (04), we will start looking at files. The most basic unit to store information.

We will learn how to work with a local file system Windows also has shell language, but we will use Unix Scripting Language.

WHY UNIX COMMANDS?

Inside Databricks, the environment is Linux.

Even if you're using Windows or macOS to access Databricks through your browser, the actual clusters (where your notebooks run) are Linux virtual machines.

So, to be very clear:

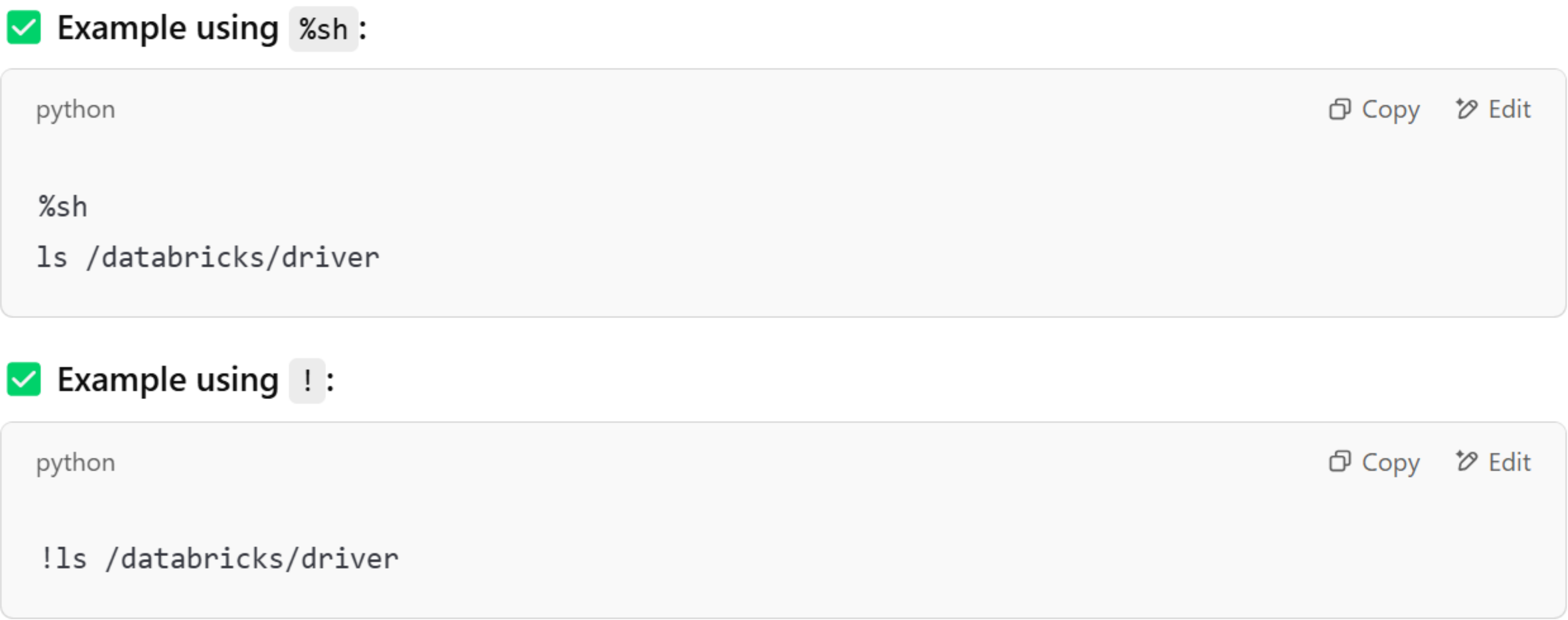
✅ Databricks backend = Linux

❌ Not Windows

This is why, for example: You can use Linux shell commands (like ls, pwd, etc.) inside Databricks notebooks. File paths follow the Linux style (/databricks/...).

HOW TO RUN SHELL COMMANDS IN DATABRICKS

Just start your command with %sh (for shell) or ! (exclamation mark).



✅ Example using `%sh`:

```
%sh
ls /databricks/driver
```

✅ Example using `!`:

```
!ls /databricks/driver
```

VERY IMPORTANT TIP

Tip: %sh is usually preferred in Databricks notebooks for multi-line shell scripts, but ! works great for quick, one-line commands.

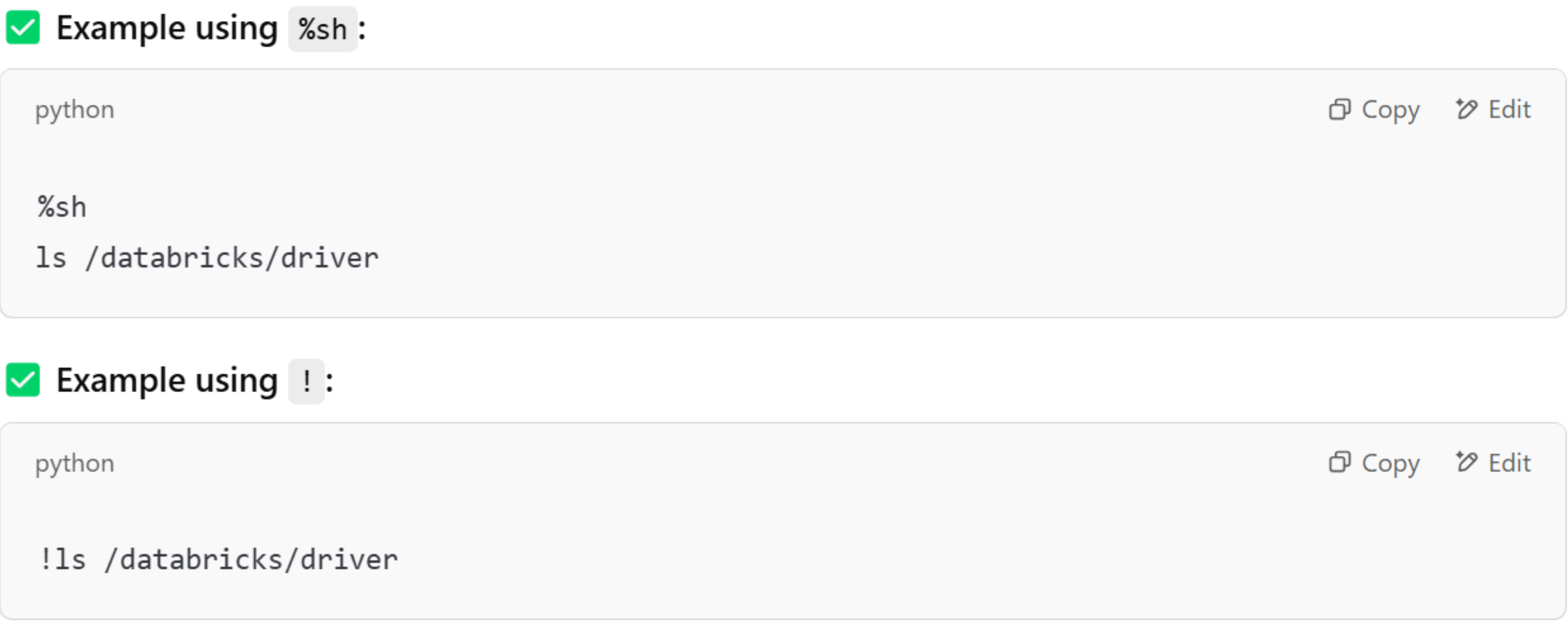
A **shell** is a program that lets you interact with your computer's operating system by typing **commands.** Think of it as a "translator" between **you** and the **system.**

```
ls        # List files
cd        # Change directory
mkdir     # Make a new folder
```

WHAT IS A SHELL?

HOW TO RUN SHELL COMMANDS IN DATABRICKS

Just start your command with %sh (for shell) or ! (exclamation mark).



✅ **Example using `%sh`:**

```
%sh
ls /databricks/driver
```

✅ **Example using `!`:**

```
!ls /databricks/driver
```

EXAMPLES OF SHELLS

Examples of shells:

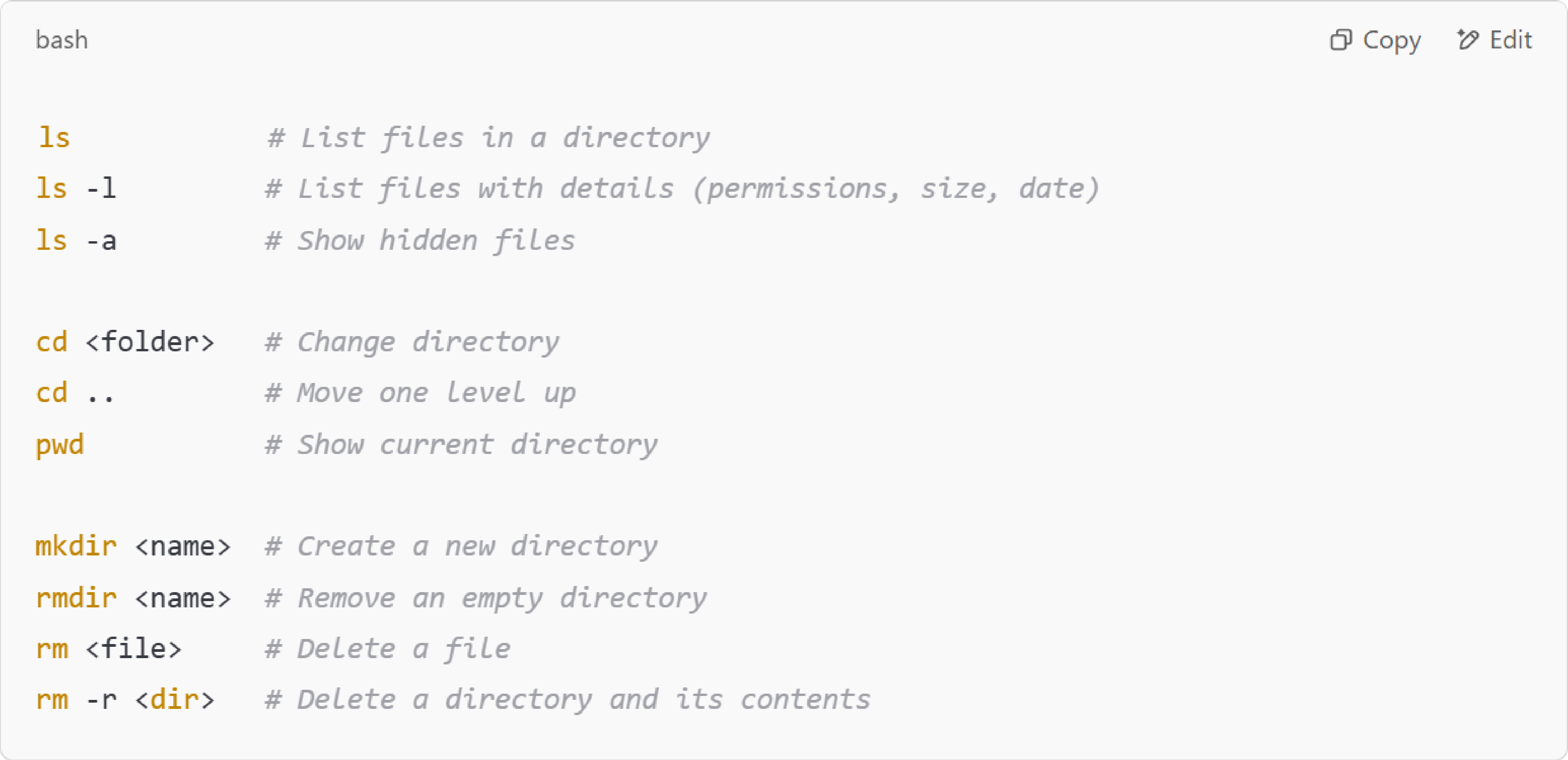
- **Bash** (common in Linux and macOS).
- **PowerShell** (common in Windows).
- **Zsh, Fish,** and others.

WHY IS SHELL USEFUL?

Why is it useful? You can manage files and folders. Run programs and scripts. Install software. Automate tasks.

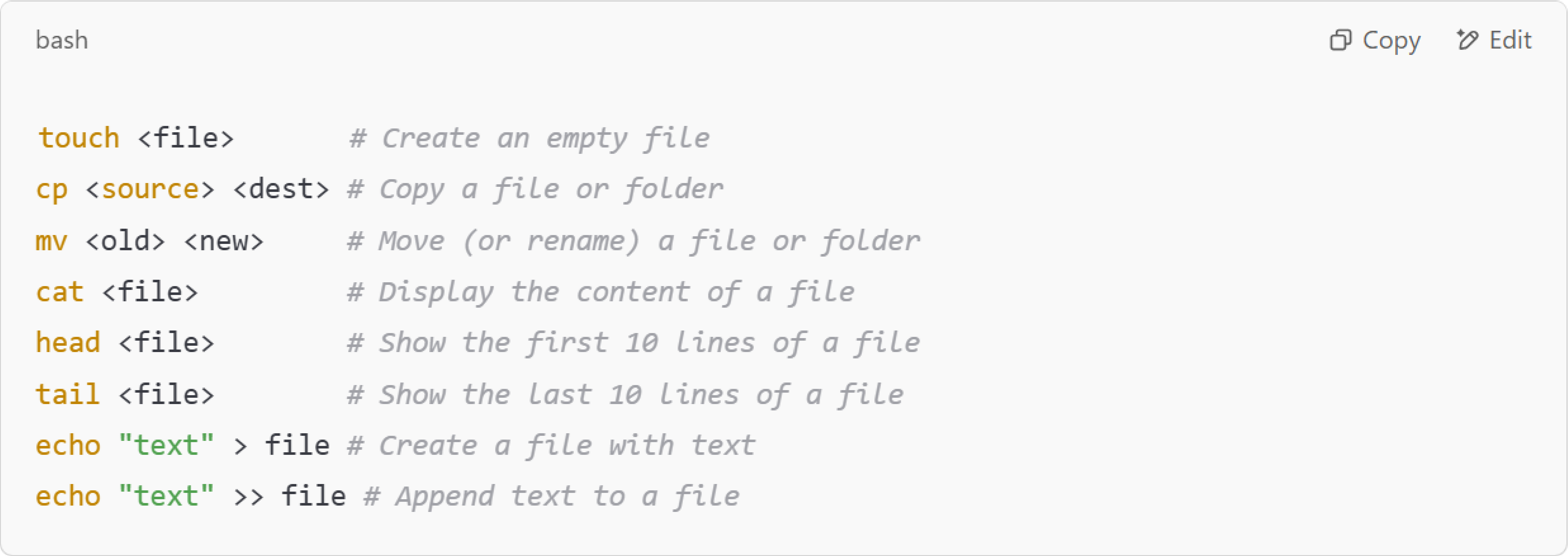
🔷 **In Databricks:** Even though Databricks is a notebook environment, it runs on Linux, so you can use Linux shell commands inside a notebook using %sh.

FILE AND DIRECTORY MANAGEMENT



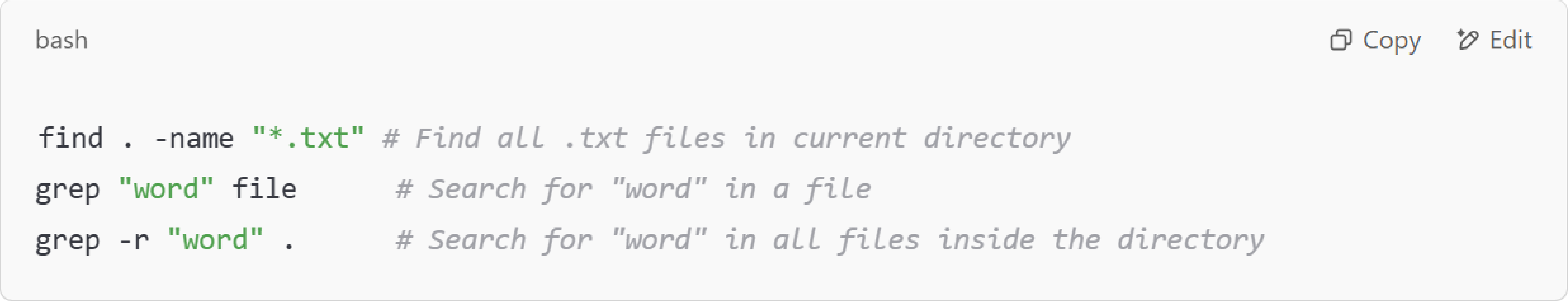
```bash
ls           # List files in a directory
ls -l        # List files with details (permissions, size, date)
ls -a        # Show hidden files

cd <folder>  # Change directory
cd ..        # Move one level up
pwd          # Show current directory

mkdir <name> # Create a new directory
rmdir <name> # Remove an empty directory
rm <file>    # Delete a file
rm -r <dir>  # Delete a directory and its contents
```

FILE OPERATIONS



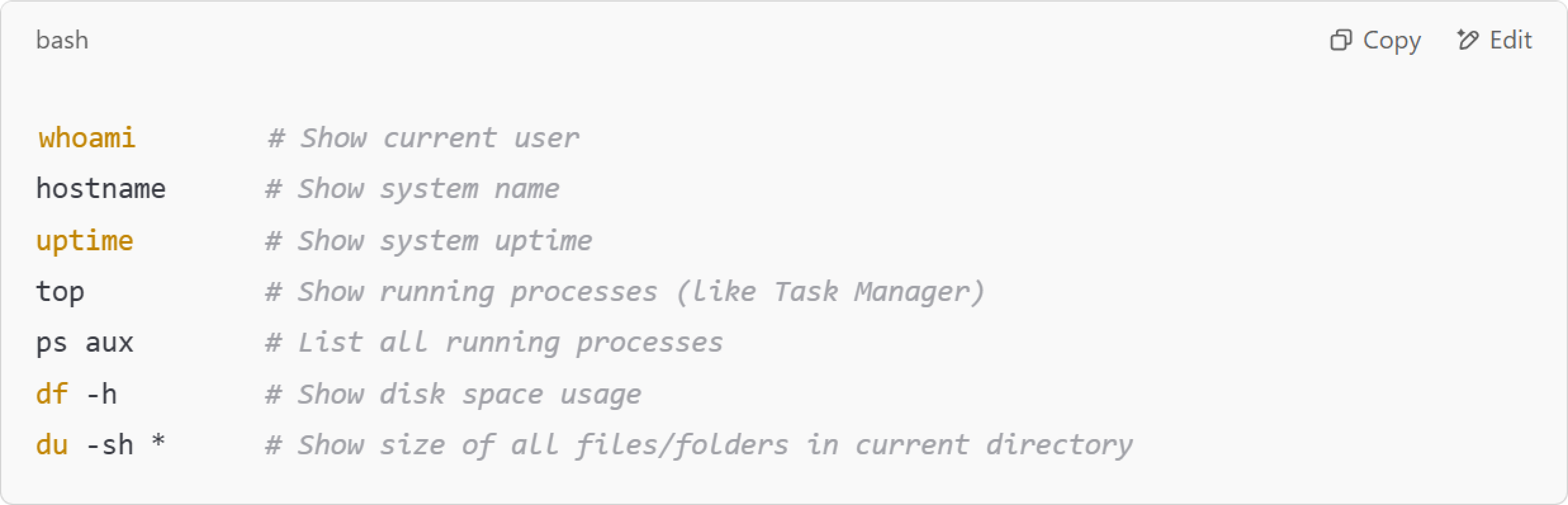
```bash
touch <file>       # Create an empty file
cp <source> <dest> # Copy a file or folder
mv <old> <new>     # Move (or rename) a file or folder
cat <file>         # Display the content of a file
head <file>        # Show the first 10 lines of a file
tail <file>        # Show the last 10 lines of a file
echo "text" > file # Create a file with text
echo "text" >> file # Append text to a file
```

SEARCHING AND FINDING FILES



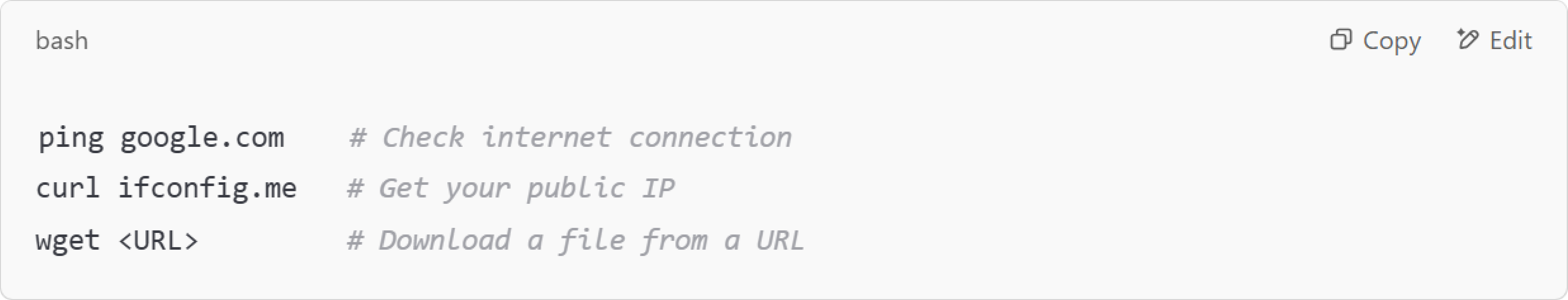
```bash
find . -name "*.txt" # Find all .txt files in current directory
grep "word" file     # Search for "word" in a file
grep -r "word" .     # Search for "word" in all files inside the directory
```

SYSTEM MONITORING



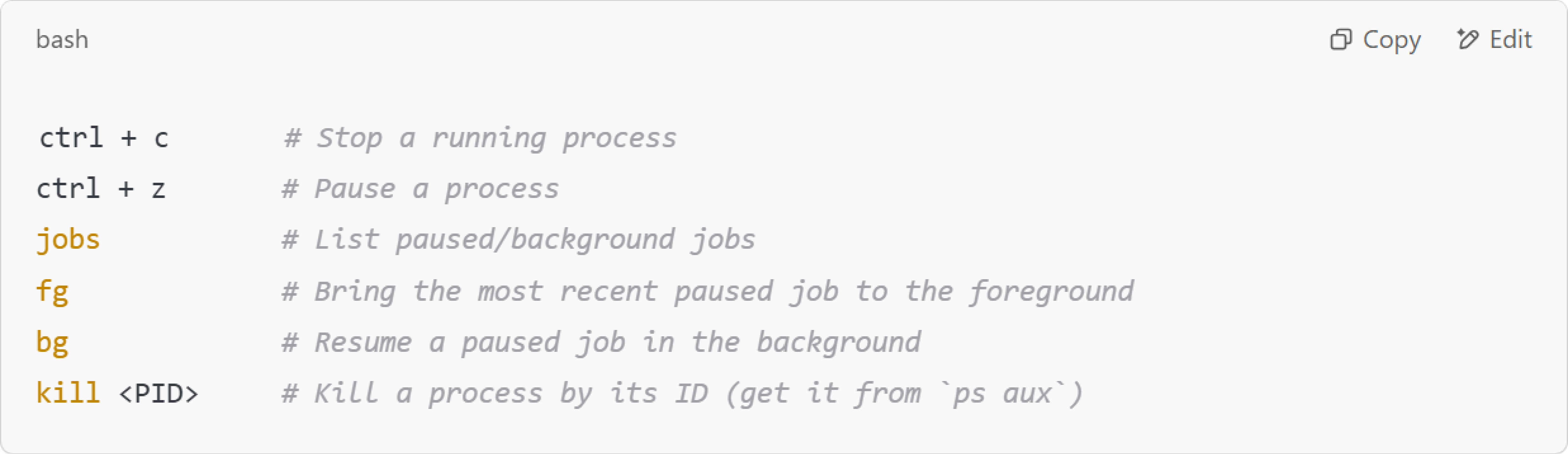
```bash
whoami        # Show current user
hostname      # Show system name
uptime        # Show system uptime
top           # Show running processes (like Task Manager)
ps aux        # List all running processes
df -h         # Show disk space usage
du -sh *      # Show size of all files/folders in current directory
```

NETWORKING



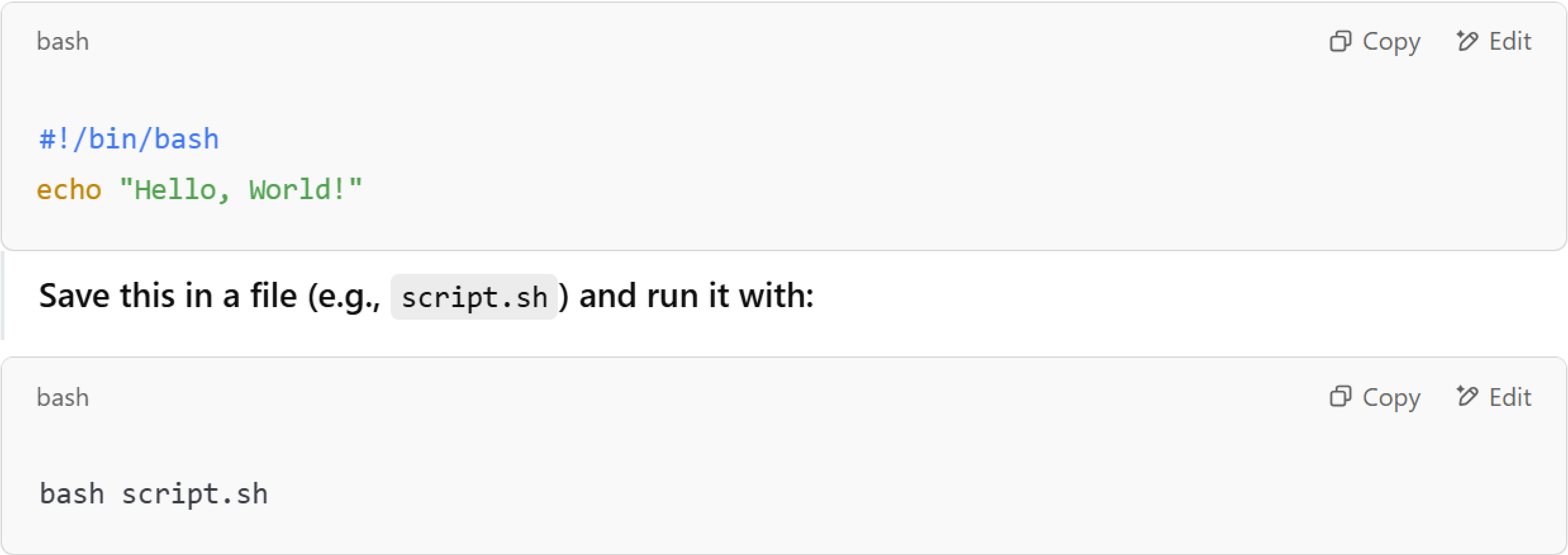
```bash
ping google.com     # Check internet connection
curl ifconfig.me    # Get your public IP
wget <URL>          # Download a file from a URL
```

PROCESS AND JOB CONTROL



```bash
ctrl + c        # Stop a running process
ctrl + z        # Pause a process
jobs            # List paused/background jobs
fg              # Bring the most recent paused job to the foreground
bg              # Resume a paused job in the background
kill <PID>      # Kill a process by its ID (get it from `ps aux`)
```

SCRIPTING AND AUTOMATION



```bash
#!/bin/bash
echo "Hello, World!"
```

Save this in a file (e.g., `script.sh`) and run it with:

```bash
bash script.sh
```

THANK YOU

Marcia Baptista

Nova IMS