

Final Notes

1. Git and Markdown A. How to clone a GitHub repository

Cloning a repository means copying a remote GitHub project to your local computer.

- Example: `bash git clone https://github.com/username/repository.git`

B. How to Use Git Commands

1. `git pull`

- **What it does:** Downloads changes from a remote repository and merges them into your local branch.
- **When to use it:** Before starting work to ensure your local copy is up to date.
- **Example:**

```
git pull origin main
```

2. `git add`

- What it does: Stages files so Git knows which changes to include in the next commit.
- When to use it: After modifying files and before committing.
- Examples:`git add file.txt`
- `git add .`

3. `git commit`

- What it does: Saves the staged changes to the local repository with a descriptive message.
- When to use it: After staging files to record changes.
- Example:`git commit -m "Updated README"`

4. `git push`

- What it does: Uploads your local commits to a remote repository.
- When to use it: After committing to share changes with others.
- Example:`git push origin main`

Typical Git Workflow

`git pull` `git add .` `git commit -m "Describe your changes"` `git push`

C. How to Write a Markdown File with Images and Proper Formatting

| Topic | Description | Command / Syntax | Example / Notes |
|-------------------|----------------------------|---|---|
| Clone GitHub Repo | Copy a remote repo locally | <code>git clone <repo_url></code> | <code>git clone https://github.com/username/repository.git</code> |

| Topic | Description | Command / Syntax | Example / Notes |
|----------------------------------|---|---|---|
| Git Pull | Update local branch with remote changes | <code>git pull origin <branch></code> | <code>git pull origin main</code> |
| Git Add | Stage files for commit | <code>git add <file></code> or <code>git add .</code> | <code>git add file.txt</code> |
| Git Commit | Save staged changes with a message | <code>git commit -m "message"</code> | <code>git commit -m "Updated README"</code> |
| Git Push | Upload commits to remote | <code>git push origin <branch></code> | <code>git push origin main</code> |
| Markdown Headings | Create titles/subtitles | <code># H1, ## H2</code> | <code># Final Notes</code> |
| Bold / Italic / Strike | Text formatting | <code>**Bold**</code> , <code>*Italic*</code> , <code>--Strike--</code> | <code>**Important**</code> , <code>*Note*</code> |
| Lists | Unordered / Ordered | <code>- Item</code> , <code>1. Item</code> | <code>- First</code> / <code>1. First</code> |
| Links / Images | Add hyperlinks or pictures | <code>[text](url), ![alt](img.png)</code> | <code>[GitHub](https://github.com), ![Logo](logo.png)</code> |
| Inline / Block Code | Highlight code | <code>`code`</code> / <code>bash ...</code> | <code>echo "Hello"</code> , <code>bash ls -la</code> |
| Redirect Output | Overwrite, append, pipe | <code>></code> , <code>>></code> , <code>>>></code> | <code>ls > files.txt</code> , <code>echo "hi" >> file.txt</code> , <code>ls grep ".txt"</code> |
| Absolute vs Relative Path | Full path vs relative | Absolute: <code>/path/to/file</code> Relative: <code>file.txt</code> | <code>touch /home/user/file.txt</code> / <code>touch docs/file.txt</code> |
| Wildcards | Match multiple files | <code>*</code> , <code>?</code> , <code>[]</code> | <code>cp *.txt /backup/</code> , <code>mv file[1-3].pdf /reports/</code> |
| Brace Expansion | Create multiple dirs/files | <code>{a,b,c}</code> | <code>mkdir -p projects/{p1,p2}/{src,bin,docs}</code> |
| Manual Pages | View command info | <code>man <command></code> | <code>man ls</code> |
| Search in man | Search for words | <code>/word</code> inside <code>man</code> | <code>/color</code> then <code>n</code> for next, <code>N</code> for previous |

| Topic | Description | Command / Syntax | Example / Notes |
|-----------------------|--|--|---|
| Hello World Script | Basic shell script | <code>#!/bin/bash</code> <code>...</code> | <code>echo "Hello, World!"</code> |
| Variables in Script | Store and reuse values | <code>var="value"</code> | <code>name="Alice", echo "Hello \$name"</code> |
| Common Linux Commands | awk, cat, cp, cut, grep, head, ls, mkdir, mv, tac, tail, touch, tr, tree | See command syntax | <code>awk '{print \$1}' file.txt, ls -l, tree -L 2</code> |
| Zip Directory | Compress folder in Debian | <code>zip -r output.zip folder/</code> | <code>zip -r my_project.zip my_project/</code> |
| Markdown → PDF | Convert Markdown file to PDF | Use VS Code extension | Right-click → Markdown PDF: Export |

Converting a Markdown File to PDF

To generate a PDF version of a Markdown file:

- 1.Open your Markdown file (e.g., README.md) in Visual Studio Code.
- 2.Right-click anywhere in the editor and select “Markdown PDF: Export (PDF)” from the context menu.
- 3.After the process completes, locate the resulting README.pdf in the same folder as the original file.

Compressing a Directory in Debian

- Before compressing, ensure that the zip utility is installed: `whereis zip sudo apt update sudo apt install zip`
- To create a zip archive of a folder:

`zip -r archive_name.zip folder_to_compress/`

`folder_to_compress/` → The directory you want to compress

`archive_name.zip` → The name of the output zip file

- Example:

`zip -r my_project.zip my_project/`

This will package the `my_project` folder into a compressed file named `my_project.zip`.

3. What are Absolute paths and relative paths? Provide examples with commands (e.g., creating a file using an absolute path)

Paths in Linux define the location of files and directories. There are two main types: absolute and relative.

Absolute Path

Represents the full path from the root directory /

Always starts with `"/`

Works regardless of your current directory

Command Example:

```
touch /home/user/documents/file.txt
```

This creates a file named file.txt in the /home/user/documents/ directory, no matter where you currently are in the filesystem.

2. Relative Path

A relative path points to a location based on your current directory.

Does not begin with /

Its meaning depends on where you currently are in the filesystem

Command Example:Here, the file file.txt is created inside the documents folder relative to /home/user.

Quick Reference: Absolute vs Relative Paths Feature Absolute Path Relative Path Starting character / Anything except / Works anywhere? Yes Only works relative to current directory Example /home/user/docs/file.txt docs/file.txt

| Feature | Absolute Path | Relative Path |
|--------------------|--------------------------|--|
| Starting character | / | Anything except / |
| Works anywhere? | Yes | Only works relative to current directory |
| Example | /home/user/docs/file.txt | docs/file.txt |

5) How to Parse (Search) for Specific Words in Manual Pages

Linux allows you to **search for specific words** inside manual pages using built-in search or by combining **man** with other commands like **grep**.

Searching Inside man Pages Using /

Open the manual page for a command: `man ls` 2.Type / followed by the word you want to search: `/color`

- 3. Press Enter to jump to the first match.
- 4. Use n to go to the next occurrence and N to go to the previous occurrence.

This allows you to quickly find specific options or keywords in any manual page.

| Operator | Function | Example |
|----------|---------------------------------|--|
| > | Overwrites output to a file | <code>ls > files.txt</code> |
| >> | Appends output to a file | <code>echo "Hello" >> files.txt</code> |
| | Pipes output to another command | <code>ls grep ".txt"</code> |

7) Appending Command Output to a File

In Linux, you can add the output of a command to the end of an existing file without overwriting its current contents using the >> operator.

- General Syntax: `command >> filename`
- Example:

```
echo "This is a new line" >> notes.txt
```

The text will be appended at the end of notes.txt rather than replacing existing content.

8) Redirecting Output with Pipes (|)

- The pipe (|) operator allows the output of one command to be sent as input to another command. This is particularly useful when processing data step by step.

- Syntax:

```
command1 | command2
```

- Example:

```
ls -l | grep ".txt"
```

- Lists only .txt files by sending the output of ls -l to grep.

9) Creating a File with Text Using echo

Wildcards are special symbols in Linux that match one or more files. They are extremely useful when you want to copy or move multiple files at once.

Common Wildcards

| Symbol | Matches | Example |
|--------|-----------------------------------|--|
| * | Any number of characters | *.txt → all .txt files |
| ? | Exactly one character | file?.txt → file1.txt, fileA.txt |
| [] | Any one character inside brackets | file[123].txt → file1.txt, file2.txt, or file3.txt |

- Copy Multiple Files
- cp *.txt /home/user/backup/ Copies all .txt files from the current directory to /home/user/backup/. cp file?.txt /home/user/backup/ Copies all files matching file1.txt, fileA.txt, etc.
- Move Multiple Files mv *.log /home/user/logs/ Moves all .log files to /home/user/logs/. Moves report1.pdf, report2.pdf, and report3.pdf to the target folder.

10) Using Brace Expansion for Directories

Brace expansion {} allows you to create multiple files or directories in a single command without typing each name individually.

Basic Syntax mkdir parent_directory/{subdir1,subdir2,subdir3}

- Creates projects/project1 projects/project2 projects/project3
2. Create nested directories
- mkdir -p projects/{project1,project2}/{src,bin,docs}

Creates:

```
projects/project1/src projects/project1/bin projects/project1/docs projects/project2/src projects/project2/bin projects/project2/docs
```

- The -p option automatically creates parent directories if they don't exist.

11. Creating a Simple "Hello World" Shell Script

Script: #!/bin/bash

This is a simple Hello World script

```
echo "Hello, World!"
```

Run in terminal: bash script.sh

Output: Hello, World!

12) Using Variables in Shell Scripts

#!/bin/bash

Simple script using variables

name="Alice" echo "Hello, \$name!" Notes:

No spaces before or after =

Access variable value using \$variable_name

13. Common Linux Commands

Below is a summary of frequently used Linux commands with definitions, syntax, and examples.

| Command | Description | Syntax / Usage | Example |
|---------|--|---|--------------------------------|
| awk | Text processing tool for pattern scanning | awk 'pattern { action }' filename | awk '{print \$1}' file.txt |
| cat | Display or concatenate files | cat [options] filename | cat file.txt |
| cp | Copy files or directories | cp [options] source destination | cp file.txt /home/user/backup/ |
| cut | Extract sections from each line | cut -d 'delimiter' -f field_number filename | cut -d, -f1 data.csv |
| grep | Search for patterns in files | grep [options] pattern filename | grep "error" log.txt |
| head | Show first part of a file | head [options] filename | head -n 5 file.txt |
| ls | List directory contents | ls [options] [directory] | ls -l |
| man | Display manual pages | man command | man ls |
| mkdir | Create directories | mkdir [options] directory_name | mkdir myfolder |
| mv | Move or rename files | mv [options] source destination | mv oldname.txt newname.txt |
| tac | Display file contents in reverse | tac filename | tac file.txt |
| tail | Display last lines of a file | tail [options] filename | tail -n 10 file.txt |
| touch | Create empty files / update timestamps | touch filename | touch newfile.txt |
| tr | Translate or delete characters | tr [options] set1 set2 | echo "hello" \ tr 'a-z' 'A-Z' |
| tree | Display directory structure in tree format | tree [options] [directory] | tree -L 2 |