DAY5 Part2 (Time 30min)

You're using the following sed command:

sed 's/am/was/g' jenkins.txt

**🔍 Let’s break it down:**

| **Part** | **Meaning** |
| --- | --- |
| sed | The stream editor command |
| 's/am/was/g' | The actual substitution command in sed |
| s | Substitution operation |
| am | The pattern to **search for** |
| was | The **replacement** text |
| g | Global flag: replace **all** occurrences in each line |
| jenkins.txt | The file where the substitution is performed (input file) |

**📄 Example in your image:**

Assuming the content of jenkins.txt was:

I am from AP, I am from nellore, I am from atmakur

After running the command, the output becomes:

I was from AP, I was from nellore, I was from atmakur

Because am was replaced with was **everywhere** in each line (due to the g flag).

⚠️ Note: This command **only prints** the output to the terminal — it doesn't change the original file unless you add the -i option:

**✅ To modify the file in place:**

sed -i 's/am/was/g' jenkins.txt

Let me know if you want to use regex, replace only specific lines, or match whole words only.

# 📌 Title: Extract Specific Line Range from a File Using `sed`

```  
sed -n '24,46p' error.txt  
```

------------------------------------------------------------------------------------

Command to extract a specific range of lines (from line 24 to line 46) from the file `error.txt` using `sed`.

Explanation:

* `sed` : stream editor used for filtering and transforming text
* `-n` : suppresses automatic printing of lines
* `'24,46p'` : tells sed to print only lines from 24 to 46
* `error.txt` : the input file from which the lines are extracted

------------------------------------------------------------------------------------

\*\*This command is useful when you want to analyze or extract a specific section of a log or text file without displaying the entire file.\*\*

# 📌 Title: Delete a Specific Line from a File Using `sed`

```  
sed '4d' kkdevops.txt  
```

------------------------------------------------------------------------------------

Command to delete a specific line (line number 4) from the file `kkdevops.txt` and display the updated content in the terminal.

Explanation:

* `sed` : stream editor used to perform basic text transformations
* `'4d'` : instructs `sed` to delete line 4
* `kkdevops.txt` : the input file being processed

------------------------------------------------------------------------------------

\*\*Note:\*\*

This command does \*\*not\*\* change the original file. It only prints the result with line 4 removed.

\*\*To update the file permanently, use the `-i` option:\*\*

```  
sed -i '4d' kkdevops.txt  
```

\*\*Always take a backup before using `-i` in case you need to recover the original data:\*\*

```  
cp kkdevops.txt kkdevops-bkp.txt  
```

# 📌 Title: View the Last Lines of a File Using `tail`

```  
tail error.txt  
```

------------------------------------------------------------------------------------

Command to display the last 10 lines of the file `error.txt`.

Explanation:

* `tail` : command used to read the end (tail) of a file
* `error.txt` : the input file whose last 10 lines are shown by default

------------------------------------------------------------------------------------

# 📌 Title: Monitor File in Real Time Using `tail -f`

```  
tail -f error.txt  
```

------------------------------------------------------------------------------------

Command to continuously monitor the end of the file `error.txt` in real-time.

Explanation:

* `-f` : 'follow' flag — it keeps the terminal open and updates the output as new lines are added to the file
* Used for live log monitoring (e.g., watching a log file as new errors appear)

------------------------------------------------------------------------------------

Use case: Helpful for debugging or tracking errors in logs as they happen without needing to reopen the file every time.

# 📘 Detailed Guide to `sed` Command with Examples

`sed` (Stream Editor) is a command-line tool used to perform basic text transformations on an input stream (a file or input from a pipeline). It's most commonly used for substitution, deletion, insertion, and other text manipulations using regular expressions.

## 📌 Syntax

```

sed [OPTIONS] 'COMMAND' [FILE...]

```

Where:

* `[OPTIONS]` - Optional flags to modify sed's behavior (`-i`, `-n`, etc.)
* `'COMMAND'` - The sed command or script
* `[FILE...]` - One or more input files

## 🔧 Common Commands and Examples

### 📌 Replace text

Command: `sed 's/apple/orange/' fruits.txt`

Explanation: Replaces the first occurrence of 'apple' with 'orange' on each line.

### 📌 Replace all occurrences

Command: `sed 's/apple/orange/g' fruits.txt`

Explanation: Replaces all occurrences of 'apple' with 'orange' on each line.

### 📌 In-place replacement

Command: `sed -i 's/old/new/g' file.txt`

Explanation: Replaces 'old' with 'new' directly in the file.

### 📌 Delete specific line

Command: `sed '3d' sample.txt`

Explanation: Deletes line number 3 from the file.

### 📌 Delete line range

Command: `sed '5,10d' file.txt`

Explanation: Deletes lines 5 through 10.

### 📌 Print specific lines

Command: `sed -n '2,4p' file.txt`

Explanation: Prints only lines 2 through 4.

### 📌 Print lines matching pattern

Command: `sed -n '/error/p' log.txt`

Explanation: Prints only lines containing the word 'error'.

### 📌 Insert line before

Command: `sed '3i\New line before 3' file.txt`

Explanation: Inserts a new line before line 3.

### 📌 Append line after

Command: `sed '3a\New line after 3' file.txt`

Explanation: Appends a new line after line 3.

### 📌 Use extended regex

Command: `sed -E 's/[0-9]+/###/g' data.txt`

Explanation: Replaces all numbers with ### using extended regex.

## 💡 Tips

* Always quote the sed command to prevent shell interpretation.
* Use `-i.bak` to make a backup of the file before in-place editing.
* You can chain multiple commands with `-e`, e.g., `sed -e 's/a/A/' -e 's/b/B/' file.txt`.
* To test changes before applying them, omit the `-i` flag.