

# PRACTICAL LINUX AND SHELL SCRIPTING SKILLS

# LINUX CLI PRACTICALS

This project catches a comprehensive hands-on session, showcasing essential Linux command-line operations for file/directory management, content manipulation, and system utilities, all vital for DevOps and administration.

# LINUX BASICS-SETUP AND FILE MANAGEMENT

Linux commands: directory creation(mkdir),navigation(cd), file creation(touch), content display (cat), and merging (cat >).Ls is used to verify changes.

```
ubuntu@ip-172-31-83-70:~$ mkdir LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS      aws_resource_tracker.sh  back      backups  hello.txt      shell-scripting-For-Devops  tws
aws_resource_tracker  aws_resources_tracker.sh  backup.sh  data     nano.6678.save  snap
ubuntu@ip-172-31-83-70:~$ cd LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cd
ubuntu@ip-172-31-83-70:~$ cd shell-scripting-For-Devops
ubuntu@ip-172-31-83-70:~/shell-scripting-For-Devops$ ls
README.md  day01  day02
ubuntu@ip-172-31-83-70:~/shell-scripting-For-Devops$ cd
ubuntu@ip-172-31-83-70:~$ cd LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ touch file1.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat>file2.txt
Hello Friends!
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt  file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file2.txt
Hello Friends!
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat>file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt  file2.txt  file3.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file2.txt
Hello Friends!
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file2.txt file3.txt>filemerge.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt  file2.txt  file3.txt  filemerge.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls *.txt
file1.txt  file2.txt  file3.txt  filemerge.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt  file2.txt  file3.txt  filemerge.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cd
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS      aws_resource_tracker.sh  back      backups  hello.txt      shell-scripting-For-Devops  tws
aws_resource_tracker  aws_resources_tracker.sh  backup.sh  data     nano.6678.save  snap
ubuntu@ip-172-31-83-70:~$ ls *.c
ls: cannot access '*.c': No such file or directory
ubuntu@ip-172-31-83-70:~$ |
```

# ADVANCED FILE OPERATIONS & ENVIRONMENT COMMANDS

This page covers further file manipulations and system information commands. It shows copying files (cp), moving files (mv), and displaying the current working directory (pwd). It also includes attempts to list specific file types (ls \*.sh, ls \*.py, ls \*.txt) and initial vi edits to file1.txt are also indicated.

```
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS      aws_resource_tracker.sh  back      backups  hello.txt      shell-scripting-For-Devops  tws
aws_resource_tracker  aws_resources_tracker.sh backup.sh  data     nano.6678.save  snap
ubuntu@ip-172-31-83-70:~$ ls *.sh
aws_resource_tracker.sh  aws_resources_tracker.sh  backup.sh
ubuntu@ip-172-31-83-70:~$ ls *.py
ls: cannot access '*.py': No such file or directory
ubuntu@ip-172-31-83-70:~$ ls *.txt
hello.txt
ubuntu@ip-172-31-83-70:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-83-70:~$ cd LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ PWD
PWD: command not found
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ pwd
/home/ubuntu/LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cpfile3.txt file4.txt
cpfile3.txt: command not found
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cp file3.txt file4.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt  file2.txt  file3.txt  file4.txt  filemerge.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file4.txt
Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cd
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS      aws_resource_tracker.sh  back      backups  hello.txt      shell-scripting-For-Devops  tws
aws_resource_tracker  aws_resources_tracker.sh backup.sh  data     nano.6678.save  snap
ubuntu@ip-172-31-83-70:~$ mv hello.txt /home/ubuntu/LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~$ cd LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt  file2.txt  file3.txt  file4.txt  filemerge.txt  hello.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file1.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file1.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file1.txt
LINE-1
LINE-2
LINE-3
LINE-4
LINE-5
LINE-6
LINE-7
```

## FILE EDITING & CONTENT UPDATE

Moving hello.txt, navigating into LINUX\_COMMANDS, and then extensively using the vi editor to add lines ("LINE-1" to "LINE-30") to file1.txt, verifying the updated content with cat.

```
ubuntu@ip-172-31-83-70:~/L ~ + ^

ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS aws_resource_tracker.sh back backups hello.txt shell-scripting-For-Devops tws
aws_resource_tracker aws_resources_tracker.sh backup.sh data nano.6678.save snap

ubuntu@ip-172-31-83-70:~$ mv hello.txt /home/ubuntu/LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~$ cd LINUX_COMMANDS
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt file2.txt file3.txt file4.txt filemerge.txt hello.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file1.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file1.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file1.txt
LINE-1
LINE-2
LINE-3
LINE-4
LINE-5
LINE-6
LINE-7
LINE-8
LINE-9
LINE-10
LINE-11
LINE-12
LINE-13
LINE-14
LINE-15
LINE-16
LINE-17
LINE-18
LINE-19
LINE-20
LINE-21
LINE-22
LINE-23
LINE-24
LINE-25
LINE-26
LINE-27
LINE-28
LINE-29
LINE-30i
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file1.txt|
```

# SPECIALIZED FILE CONTENT DISPLAY

Demonstrates commands for specific file content viewing: head (first 10 lines), tail (last 10 lines), and tac (entire content reversed), all applied to file1.txt.f body text

```
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ head file1.txt
LINE-1
LINE-2
LINE-3
LINE-4
LINE-5
LINE-6
LINE-7
LINE-8
LINE-9
LINE-10
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ tail file1.txt
LINE-21
LINE-22
LINE-23
LINE-24
LINE-25
LINE-26
LINE-27
LINE-28
LINE-29
LINE-30i
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ tac file1.txt
LINE-30i
LINE-29
LINE-28
LINE-27
LINE-26
LINE-25
LINE-24
LINE-23
LINE-22
LINE-21
LINE-20
LINE-19
LINE-18
LINE-17
LINE-16
LINE-15
LINE-14
LINE-13
```

## FULL REVERSED FILE DISPLAY

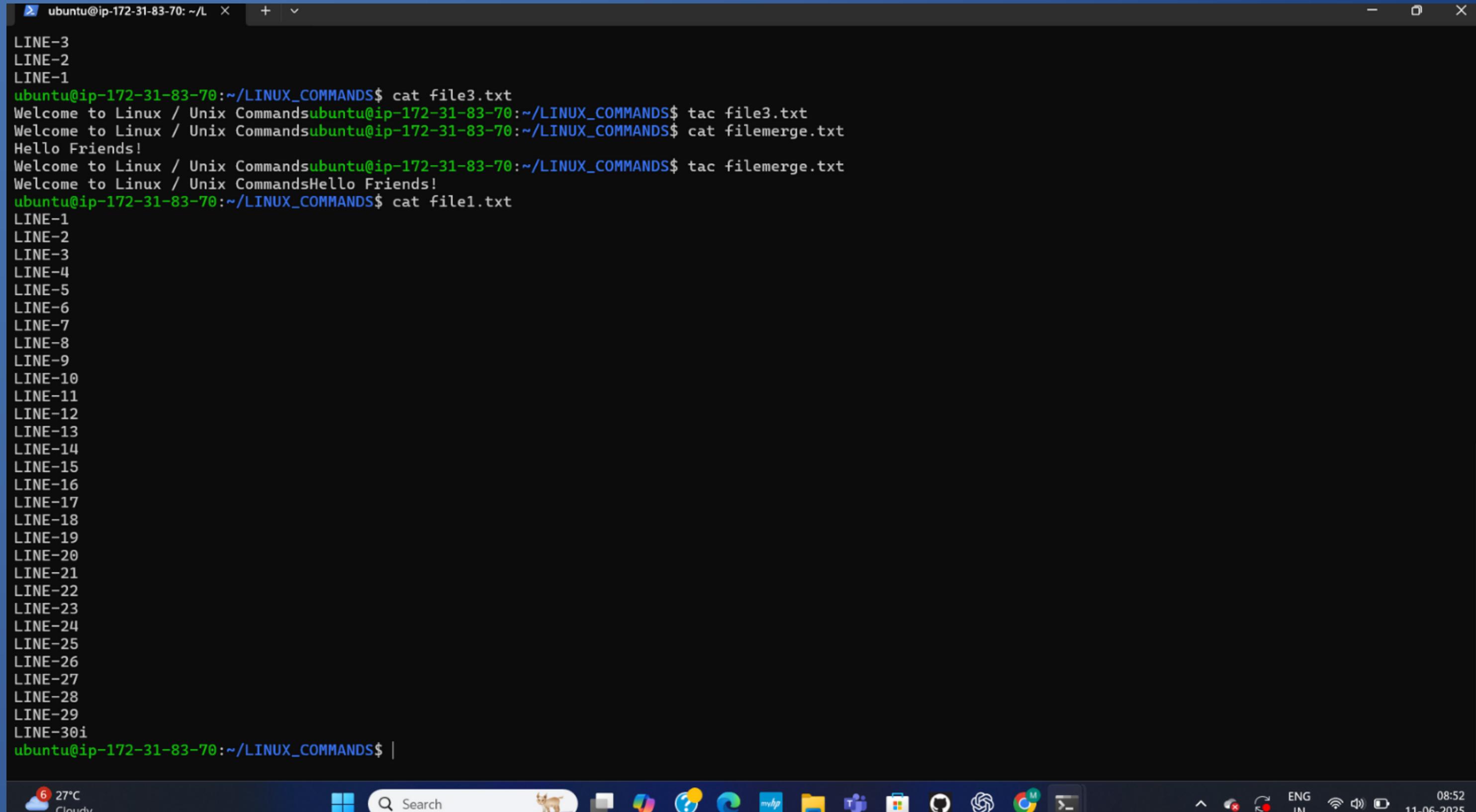
The tac command output, displaying the complete content of file1.txt in reverse order, from "LINE-30" back to "LINE-1".

```
ubuntu@ip-172-31-83-70:~/L X + ▾ - □ X
LINE-22
LINE-23
LINE-24
LINE-25
LINE-26
LINE-27
LINE-28
LINE-29
LINE-30i
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ tac file1.txt
LINE-30i
LINE-29
LINE-28
LINE-27
LINE-26
LINE-25
LINE-24
LINE-23
LINE-22
LINE-21
LINE-20
LINE-19
LINE-18
LINE-17
LINE-16
LINE-15
LINE-14
LINE-13
LINE-12
LINE-11
LINE-10
LINE-9
LINE-8
LINE-7
LINE-6
LINE-5
LINE-4
LINE-3
LINE-2
LINE-1
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

## TAC ON OTHER FILES & FULL FILE1.TXT VIEW

This Further illustrates the tac command on file3.txt and filemerge.txt, showing how it reverses their content.  
The cat command is also used to display the full content of file1.txt.

```
ubuntu@ip-172-31-83-70:~/L X + v - o X
LINE-3
LINE-2
LINE-1
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ tac file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat filemerge.txt
Hello Friends!
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ tac filemerge.txt
Welcome to Linux / Unix CommandsHello Friends!
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file1.txt
LINE-1
LINE-2
LINE-3
LINE-4
LINE-5
LINE-6
LINE-7
LINE-8
LINE-9
LINE-10
LINE-11
LINE-12
LINE-13
LINE-14
LINE-15
LINE-16
LINE-17
LINE-18
LINE-19
LINE-20
LINE-21
LINE-22
LINE-23
LINE-24
LINE-25
LINE-26
LINE-27
LINE-28
LINE-29
LINE-30i
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```



The screenshot shows a terminal window on an Ubuntu desktop. The terminal displays a sequence of commands and their outputs. It starts with 'cat file3.txt' showing three lines of text: 'LINE-3', 'LINE-2', and 'LINE-1'. Then it runs 'tac file3.txt' which prints the same three lines in reverse order: 'LINE-1', 'LINE-2', and 'LINE-3'. Next, it runs 'cat filemerge.txt' which prints 'Hello Friends!' followed by 'Welcome to Linux / Unix Commands'. Finally, it runs 'tac filemerge.txt' which prints 'Hello Friends!' followed by 'Welcome to Linux / Unix Commands'. Below the terminal, the desktop environment includes a weather icon (27°C), a search bar, and various application icons in the dock.

# USER AND SEARCH UTILITIES

In this it introduces commands for system information and content searching. It shows more for paginated file viewing of file1.txt, id for user and group information (uid, gid, groups of ubuntu), and grep for searching specific patterns ("Linux", "o") within file3.txt.

```
ubuntu@ip-172-31-83-70:~/L X + ▾
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ more file1.txt
LINE-1
LINE-2
LINE-3
LINE-4
LINE-5
LINE-6
LINE-7
LINE-8
LINE-9
LINE-10
LINE-11
LINE-12
LINE-13
LINE-14
LINE-15
LINE-16
LINE-17
LINE-18
LINE-19
LINE-20
LINE-21
LINE-22
LINE-23
LINE-24
LINE-25
LINE-26
LINE-27
LINE-28
LINE-29
LINE-30i
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ id
uid=1000(ubuntu) gid=1000(ubuntu) groups=1000(ubuntu),4(adm),24(cdrom),27(sudo),30(dip),105(lxd)
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ grep Linux file3.txt
Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ grep o file3.txt
Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file2.txt
Hello Friends!
```

# FILE COMPARISON AND NETWORK TESTING

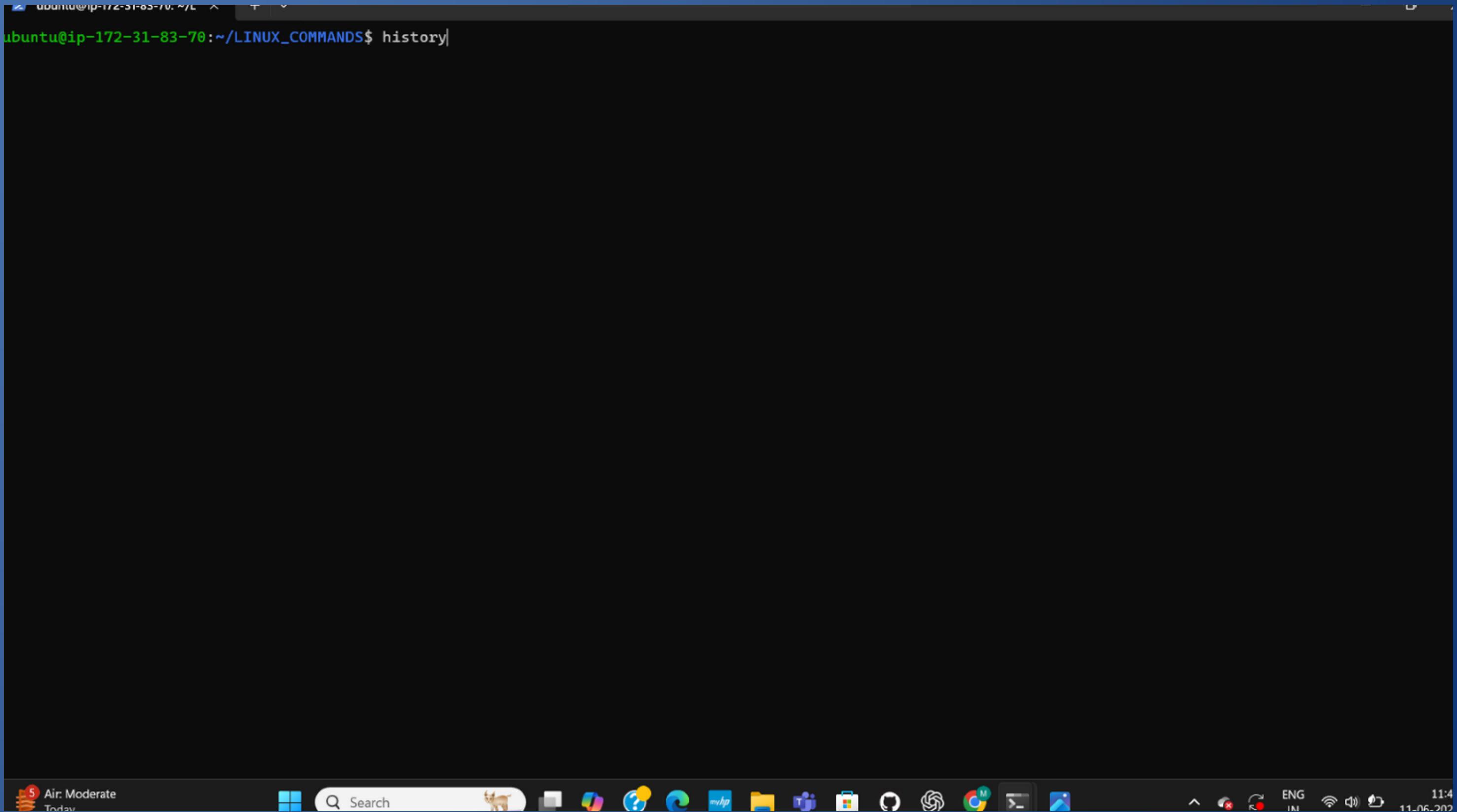
This focuses on comparing file contents (diff file2.txt file3.txt and diff file2.txt filemerge.txt) and network diagnostics (ping google.com), showing differences between files and successful ping responses.

```
ubuntu@ip-172-31-83-70:~/L X + v - o X
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ diff file2.txt file3.txt
1c1
< Hello Friends!
---
> Welcome to Linux / Unix Commands
\ No newline at end of file
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat filemerge.txt
Hello Friends!
Welcome to Linux / Unix Commandsubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ diff file2.txt filemerge.txt
1a2
> Welcome to Linux / Unix Commands
\ No newline at end of file
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file2.txt
Hello Friends!
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ping google.com
PING google.com (172.253.115.113) 56(84) bytes of data.
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=1 ttl=106 time=2.55 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=2 ttl=106 time=2.11 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=3 ttl=106 time=2.23 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=4 ttl=106 time=2.09 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=5 ttl=106 time=2.18 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=6 ttl=106 time=2.32 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=7 ttl=106 time=2.07 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=8 ttl=106 time=2.10 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=9 ttl=106 time=2.41 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=10 ttl=106 time=2.56 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=11 ttl=106 time=2.92 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=12 ttl=106 time=2.19 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=13 ttl=106 time=2.25 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=14 ttl=106 time=2.19 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=15 ttl=106 time=2.28 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=16 ttl=106 time=2.14 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=17 ttl=106 time=2.07 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=18 ttl=106 time=2.38 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=19 ttl=106 time=2.53 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=20 ttl=106 time=2.52 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=21 ttl=106 time=2.09 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=22 ttl=106 time=2.08 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=23 ttl=106 time=2.30 ms
64 bytes from bg-in-f113.1e100.net (172.253.115.113): icmp_seq=24 ttl=106 time=4.23 ms

10
```

## COMMAND HISTORY INITIATION

It shows the execution of the history command, indicating that a list of previously entered commands is being requested and will be displayed.



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following text:

```
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ history|
```

The terminal window is positioned above a dock containing various application icons. The desktop background is a light blue gradient.

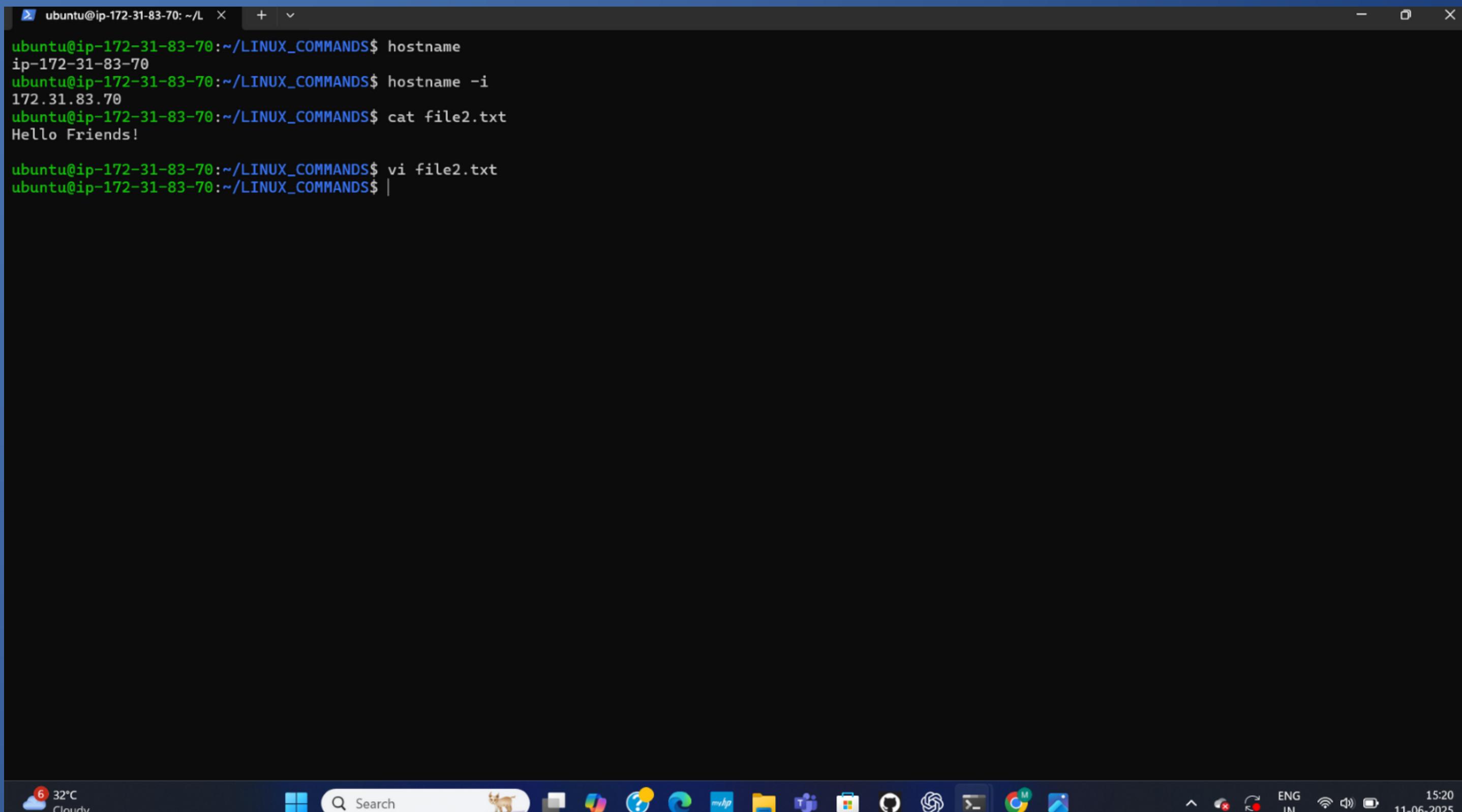
## COMMAND HISTORY OUTPUT

It presents a detailed output of the history command, listing numerous commands executed in the session with their corresponding command numbers. This includes commands like cd, pwd, cp, ls, cat, vi, head, tail, tac, clear, ping, and history itself.

```
ubuntu@ip-172-31-83-70: ~/L X + | - 
655 cd LINUX_COMMANDS
656 PWD
657 pwd
658 cpfile3.txt file4.txt
659 cp file3.txt file4.txt
660 ls
661 cat file3.txt
662 cat file4.txt
663 cd
664 ls
665 mv hello.txt /home/ubuntu/LINUX_COMMANDS
666 cd LINUX_COMMANDS
667 ls
668 cat file1.txt
669 vi file1.txt
670 cat file1.txt
671 head file1.txt
672 tial file1.txt
673 tia1 file1.txt
674 clear
675 head file1.txt
676 tail file1.txt
677 tac file1.txt
678 cat file3.txt
679 tac file3.txt
680 cat filemerge.txt
681 tac filemerge.txt
682 cat file1.txt
683 clear
684 ls
685 cd LINUX_COMMANDS
686 LS
687 clear
688 ping google.com
689 clear
690 history
691 clear
692 history
693 clear
694 history
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

# HOSTNAME INFORMATION AND FILE VIEWING

This displays commands related to system identification and basic file viewing. It shows hostname to get the system's name (ip-172-31-83-70) and hostname -i for its IP address (172.31.83.70). Additionally, it shows cat file2.txt to display the content "Hello Friends!" and vi file2.txt for editing.



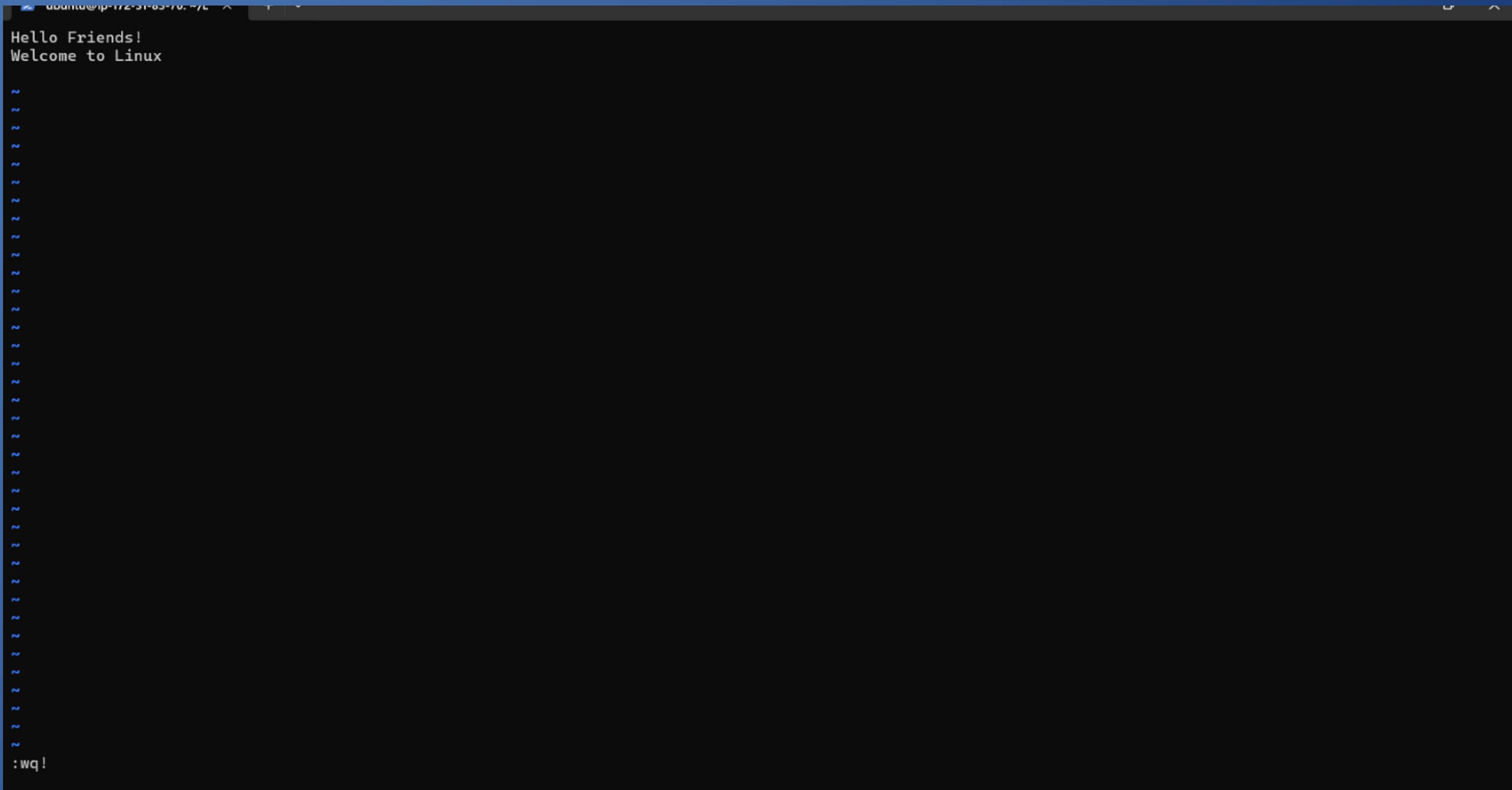
The screenshot shows a terminal window with a black background and white text. At the top, the title bar displays the session information: "ubuntu@ip-172-31-83-70: ~/L". Below the title bar, the terminal window contains the following command history:

```
ubuntu@ip-172-31-83-70:~/L$ hostname  
ip-172-31-83-70  
ubuntu@ip-172-31-83-70:~/L$ hostname -i  
172.31.83.70  
ubuntu@ip-172-31-83-70:~/L$ cat file2.txt  
Hello Friends!  
  
ubuntu@ip-172-31-83-70:~/L$ vi file2.txt  
ubuntu@ip-172-31-83-70:~/L$ |
```

At the bottom of the terminal window, there is a standard Linux desktop taskbar with various icons for applications like File Explorer, Microsoft Edge, and Google Chrome. The system tray on the right side of the taskbar shows the date (11-06-2025), time (15:20), battery level (ENG IN), signal strength, and a weather widget indicating 32°C and Cloudy conditions.

## VI EDITOR VIEW OF FILE2.TXT

it shows a visual representation of file2.txt within the vi editor, displaying its initial content "Hello Friends!" and "Welcome to Linux".



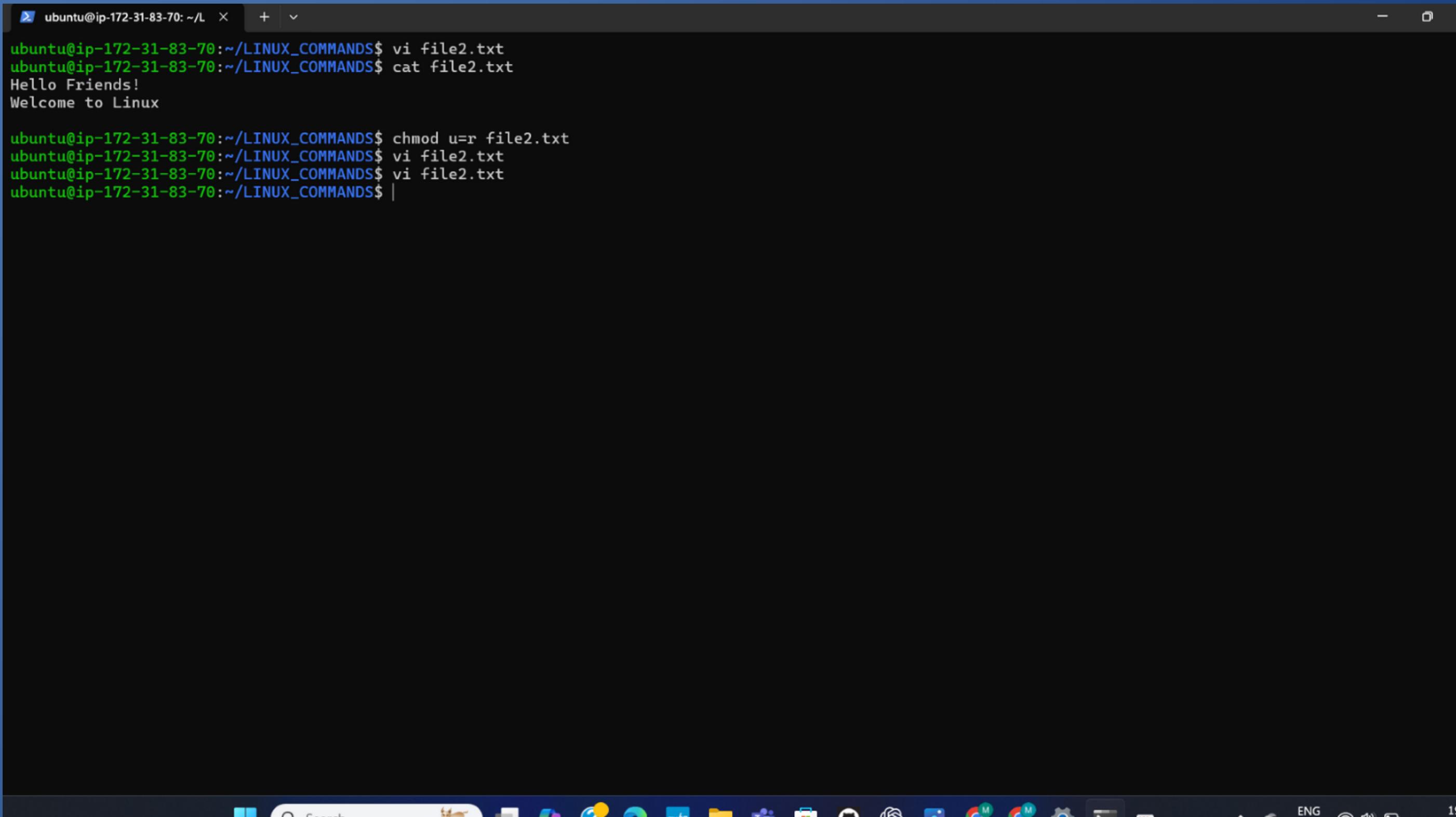
The screenshot shows a terminal window with the title bar "Ubuntu@ip-172-51-85-70: ~". The main area displays the contents of a file named "file2.txt". The text reads:

```
Hello Friends!
Welcome to Linux
```

Below the text, there is a vertical scroll bar on the left side of the terminal window. At the bottom of the screen, the command ":wq!" is visible, indicating the user's intention to write changes and quit the editor.

## FILE PERMISSIONS (ATTEMPTED) AND VI EXIT

This shows an attempted chmod command (chmod user), which is likely incorrect syntax for changing file permissions. It then shows vi file2.txt and cat file2.txt, confirming its content.



The screenshot shows a terminal window titled "ubuntu@ip-172-31-83-70: ~/L". The terminal history is as follows:

```
ubuntu@ip-172-31-83-70:~/L ~/+/- 
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file2.txt
Hello Friends!
Welcome to Linux

ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ chmod u=r file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

The terminal window is set against a dark blue background. The bottom of the screen shows a standard Linux desktop interface with icons for various applications like a browser, file manager, and system settings. The status bar at the bottom right indicates "ENG" and "19".

# VI EDITOR WITH READ-ONLY NOTE

This displays file2.txt again within the vi editor, showing its content ("Hello Friends!", "Welcome to Linux", "Thank you") and a "readonly" note at the bottom.

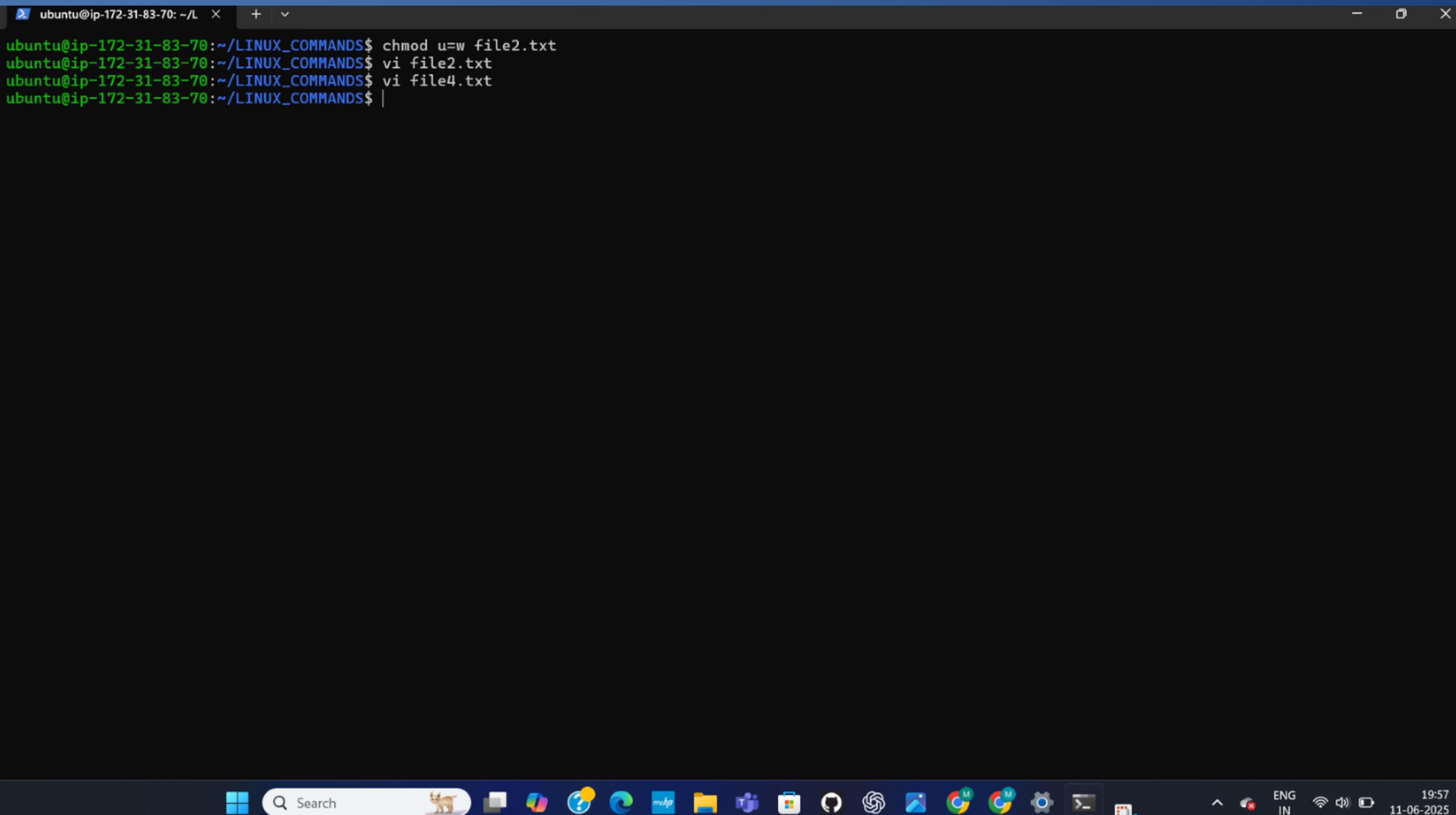
A screenshot of a Linux terminal window titled "ubuntu@ip-172-31-83-70: ~/L". The window contains the following text:

```
Hello Friends!
Welcome to Linux
Thank you
```

The status bar at the bottom shows the file path "/file2.txt [readonly] 4L, 43B", the line number "3, 9", and the word "All". The desktop environment's taskbar is visible at the bottom, featuring icons for search, file explorer, and various system applications.

# CHMOD CORRECTION & VI RETURN

This shows the corrected chmod command (chmod u+w file2.txt), followed by re-opening file2.txt in vi, and then cat file2.txt.



The screenshot shows a terminal window on a Windows desktop. The terminal has a dark background and white text. It displays the following commands:

```
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ chmod u=w file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file2.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file4.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

The desktop taskbar at the bottom shows various application icons, including Microsoft Edge, File Explorer, and Google Chrome. The system tray indicates the date as 11-06-2025 and the time as 19:57. Language settings show "ENG IN".

# VI EDITOR FOR FILE4.TXT

This shows file4.txt opened in the vi editor, displaying its content "Welcome to Linux / Unix Commands", with a "readonly" note at the bottom.

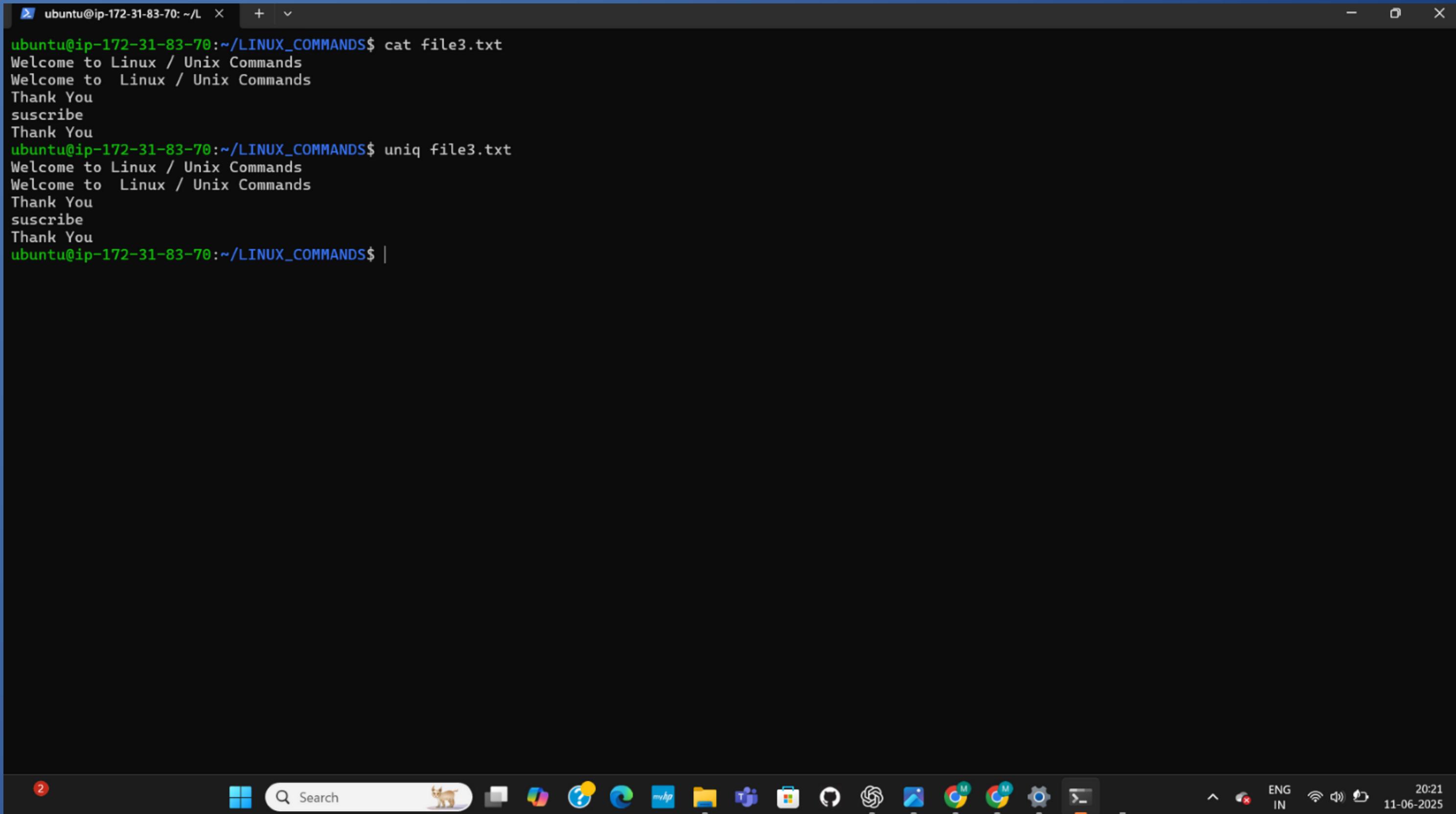
## FILE CONTENT AND NL COMMAND

This shows vi file4.txt, ls, cat file4.txt, and nl file4.txt. The nl file1.txt command is also shown, displaying file1.txt with line numbers from 1 to 30.

```
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file4.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ vi file5.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ ls
file1.txt file2.txt file3.txt file4.txt filemerge.txt hello.txt
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file4.txt
Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ nl file4.txt
    1 Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ nl file3.txt
    1 Welcome to Linux / Unix Commands
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ nl file1.txt
    1 LINE-1
    2 LINE-2
    3 LINE-3
    4 LINE-4
    5 LINE-5
    6 LINE-6
    7 LINE-7
    8 LINE-8
    9 LINE-9
   10 LINE-10
   11 LINE-11
   12 LINE-12
   13 LINE-13
   14 LINE-14
   15 LINE-15
   16 LINE-16
   17 LINE-17
   18 LINE-18
   19 LINE-19
   20 LINE-20
   21 LINE-21
   22 LINE-22
   23 LINE-23
   24 LINE-24
   25 LINE-25
   26 LINE-26
   27 LINE-27
   28 LINE-28
   29 LINE-29
   30 LINE-30i
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

# WORD COUNT AND FILE EDITING

This wc file3.txt to count words, lines, and characters in file3.txt. An attempt to cat file3aa results in "No such file or directory". wc file1.txt is also shown, giving character and line counts. The page ends with vi file1.txt.



```
ubuntu@ip-172-31-83-70:~/L X + v
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commands
Welcome to Linux / Unix Commands
Thank You
suscribe
Thank You
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ uniq file3.txt
Welcome to Linux / Unix Commands
Welcome to Linux / Unix Commands
Thank You
suscribe
Thank You
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

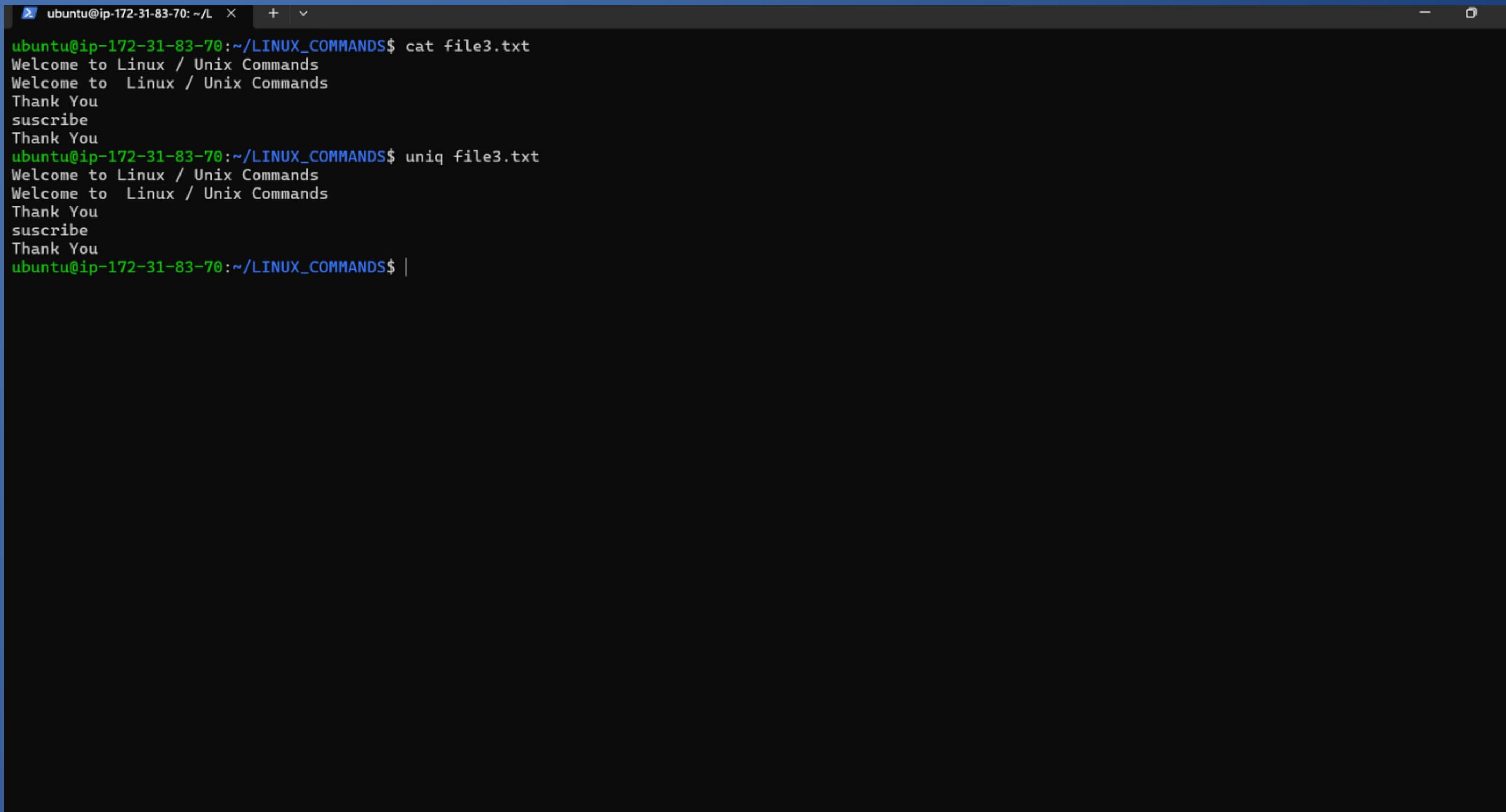
The screenshot shows a terminal window with a dark background and light-colored text. It displays a command-line session where the user runs 'cat file3.txt' followed by 'uniq file3.txt'. The output shows the original file content followed by a blank line. Below the terminal is a Windows-style taskbar with various icons for applications like File Explorer, Microsoft Edge, and Google Chrome. The system tray shows the date as 11-06-2025 and the time as 20:21. The language setting is ENG IN.

# VI EDITOR WITH ADDITIONAL LINES

This displays file1.txt within the vi editor, showing content "Welcome to Linux / Unix Commands", "Thank You", and "suscribe".

# UNIQ COMMAND DEMONSTRATION

It illustrates the uniq command on file3.txt, which is used to filter out or report repeated lines.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the command prompt: `ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$`. Below the prompt, the user runs two commands: `cat file3.txt` and `uniq file3.txt`. The output of `cat file3.txt` shows the original content of the file, which includes several repeated lines: "Welcome to Linux / Unix Commands", "Thank You", "suscribe", and "Thank You". The output of `uniq file3.txt` shows the content after the uniq command has been applied, where the repeated lines have been removed, leaving only the unique lines: "Welcome to Linux / Unix Commands", "Thank You", "suscribe", and "Thank You". The terminal window also features standard window control buttons (minimize, maximize, close) at the top right.

```
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ cat file3.txt
Welcome to Linux / Unix Commands
Welcome to Linux / Unix Commands
Thank You
suscribe
Thank You
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ uniq file3.txt
Welcome to Linux / Unix Commands
Thank You
suscribe
Thank You
ubuntu@ip-172-31-83-70:~/LINUX_COMMANDS$ |
```

# DIRECTORY OPERATIONS AND LISTING

This shows attempts to remove directories using rmdir (which fail with "Directory not empty"). ls commands are used to list directory contents after these operations. ro back and ro snap commands are also shown, likely related to deleting files or directories.

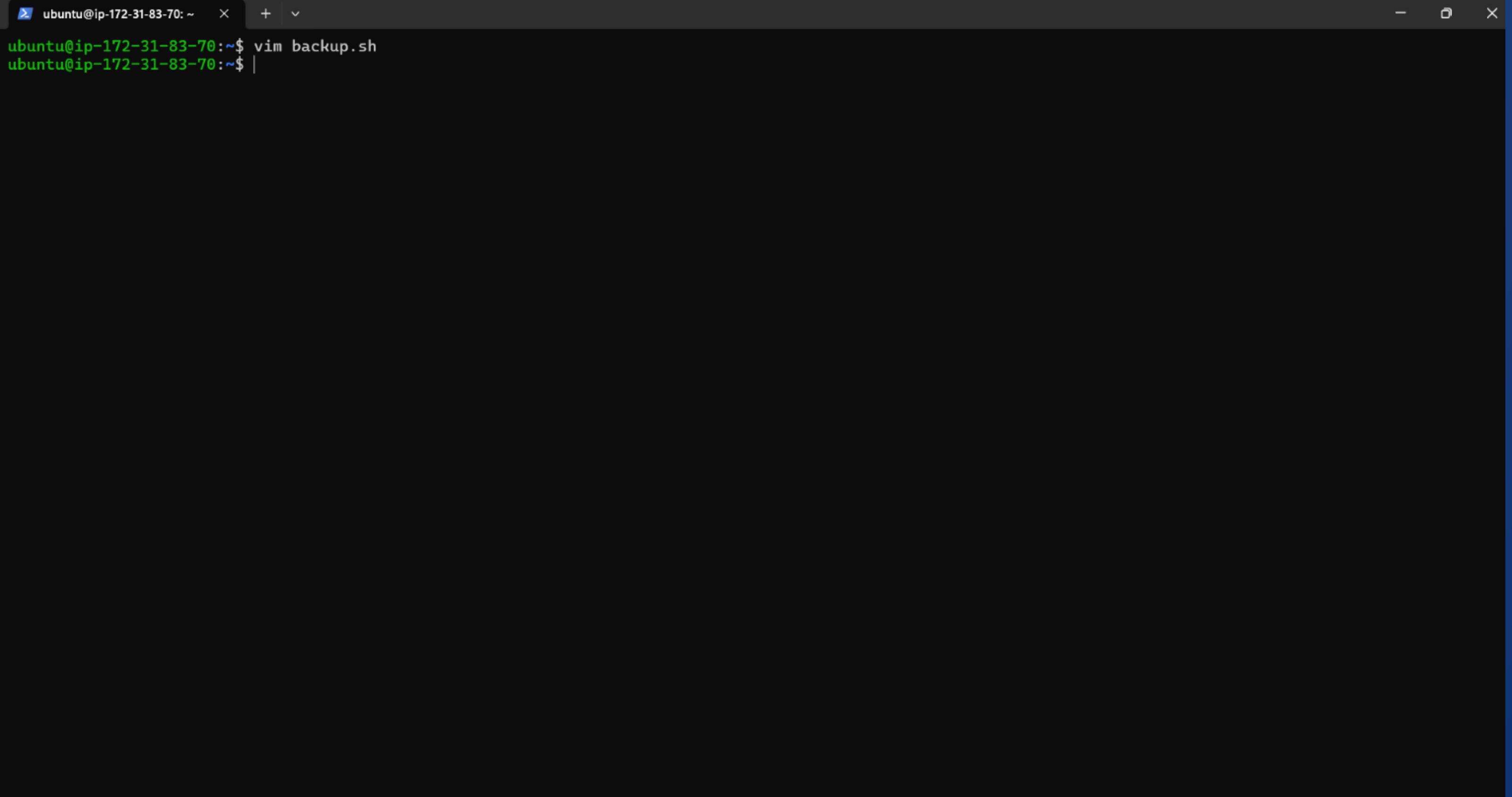
```
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS aws_resource_tracker aws_resources_tracker.sh backup.sh file2.txt nano.6678.save snap
Linux_commands aws_resource_tracker.sh back backups file4.txt shell-scripting-For-Devops tws
ubuntu@ip-172-31-83-70:~$ ls snap
aws-cli
ubuntu@ip-172-31-83-70:~$ rmmdir snap
rmmdir: failed to remove 'snap': Directory not empty
ubuntu@ip-172-31-83-70:~$ rmmdir backups
rmmdir: failed to remove 'backups': Directory not empty
ubuntu@ip-172-31-83-70:~$ rmmdir tws
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS aws_resource_tracker aws_resources_tracker.sh backup.sh file2.txt nano.6678.save snap
Linux_commands aws_resource_tracker.sh back backups file4.txt shell-scripting-For-Devops
ubuntu@ip-172-31-83-70:~$ ls snap
aws-cli
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS aws_resource_tracker aws_resources_tracker.sh backup.sh file2.txt nano.6678.save snap
Linux_commands aws_resource_tracker.sh back backups file4.txt shell-scripting-For-Devops
ubuntu@ip-172-31-83-70:~$ rm back
ubuntu@ip-172-31-83-70:~$ ls
LINUX_COMMANDS aws_resource_tracker aws_resources_tracker.sh backups file4.txt shell-scripting-For-Devops
Linux_commands aws_resource_tracker.sh backup.sh file2.txt nano.6678.save snap
ubuntu@ip-172-31-83-70:~$ |
```

# AUTOMATED BACKUP SCRIPT

This project focuses on the development and automation of a shell script, backup.sh, designed to create and manage system backups with a 5-day rotation policy.

## INITIAL SESSION DETAILS

I HAVE began a session on ip-172-31-83-70. The primary activity shown is initiating vim to edit backup.sh.



A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70: ~". The window contains the following text:

```
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ |
```

# SCRIPT INTRODUCTION AND

This page introduces `backup.sh` as a bash script designed for a 5-day backup rotation. It specifies the script's usage:  
**USAGE**  
`/backup.sh <path to your source> <path to backup folder>`. A `display_usage` function is defined to print this message, and the script includes a check (`if [ $# -eq ]`) to call `display_usage` if no arguments are provided.

# INITIAL EXECUTION & DATE FORMATTING

This page shows the user's current directory (/home/ubuntu) and lists its contents, including backup.sh. When backup.sh is executed without arguments, it correctly outputs its usage message. The user also demonstrates experimenting with the date command to format timestamps (e.g., +%Y-%m-%d-%H-%M-%S).

```
ubuntu@ip-172-31-83-70:~$ ls
aws_resource_tracker aws_resource_tracker.sh aws_resources_tracker.sh backup.sh backups data hello.txt shell-scripting-For-Devops snap tws
ubuntu@ip-172-31-83-70:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-83-70:~$ ./backup.sh
Usage: ./backup.sh <path to your source> <path to backup folder>
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backup
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ date
Fri Jun  6 03:28:43 UTC 2025
ubuntu@ip-172-31-83-70:~$ date '+%Y-%m-%d-%H-%M-%S'
2025-06-06-03-30-35
ubuntu@ip-172-31-83-70:~$ date '+%Y-%m-%d-%H-%M-%S'
2025-06-06-03-30-49
ubuntu@ip-172-31-83-70:~$ date '+%Y-%m-%d-%H-%M-%S'
2025-06-06-03-31-14
ubuntu@ip-172-31-83-70:~$ vim backup.sh|
```

# IMPLEMENTING BACKUP CREATION FUNCTION

This page shows the backup.sh script being updated to define variables for source\_dir (\$1) and backup\_dir (\$2). A timestamp variable is also set using date +%Y-%m-%d-%H-%M-%S. The core create\_backup function is introduced, which is responsible for zipping the source\_dir into a timestamped file named backup\_\${timestamp}.zip within the backup\_dir. The function call create\_backup is also shown.

```
#!/bin/bash

<< readme

This is a script for backup with 5 day rotation

Usage:
./backup.sh <path to your source> <path to backup folder>
readme

function display_usage {

    echo "Usage: ./backup.sh <path to your source> <path to backup folder>"
}

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {

    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}"

    echo "backup generated successfully for ${timestamp}"
}

create_backup

~  
~  
~  
~  
~  
~  
~  
:wq|
```

# ADDING BACKUP SUCCESS CHECK

IN THIS the create\_backup function being updated to include an error check. Specifically, an if condition (if [ \$? -eq ]); is added immediately after the zip command. This condition checks the exit status of the zip command to determine if it executed successfully. If successful, the script echoes a message: "backup generated successfully for \${timestamp}".

```
ubuntu@ip-172-31-83-70: ~ + v - X
#!/bin/bash

<< readme

This is a script for backup with 5 day rotation

Usage:
./backup.sh <path to your source> <path to backup folder>
readme

function display_usage {

    echo "Usage: ./backup.sh <path to your source> <path to backup folder>"
}

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {

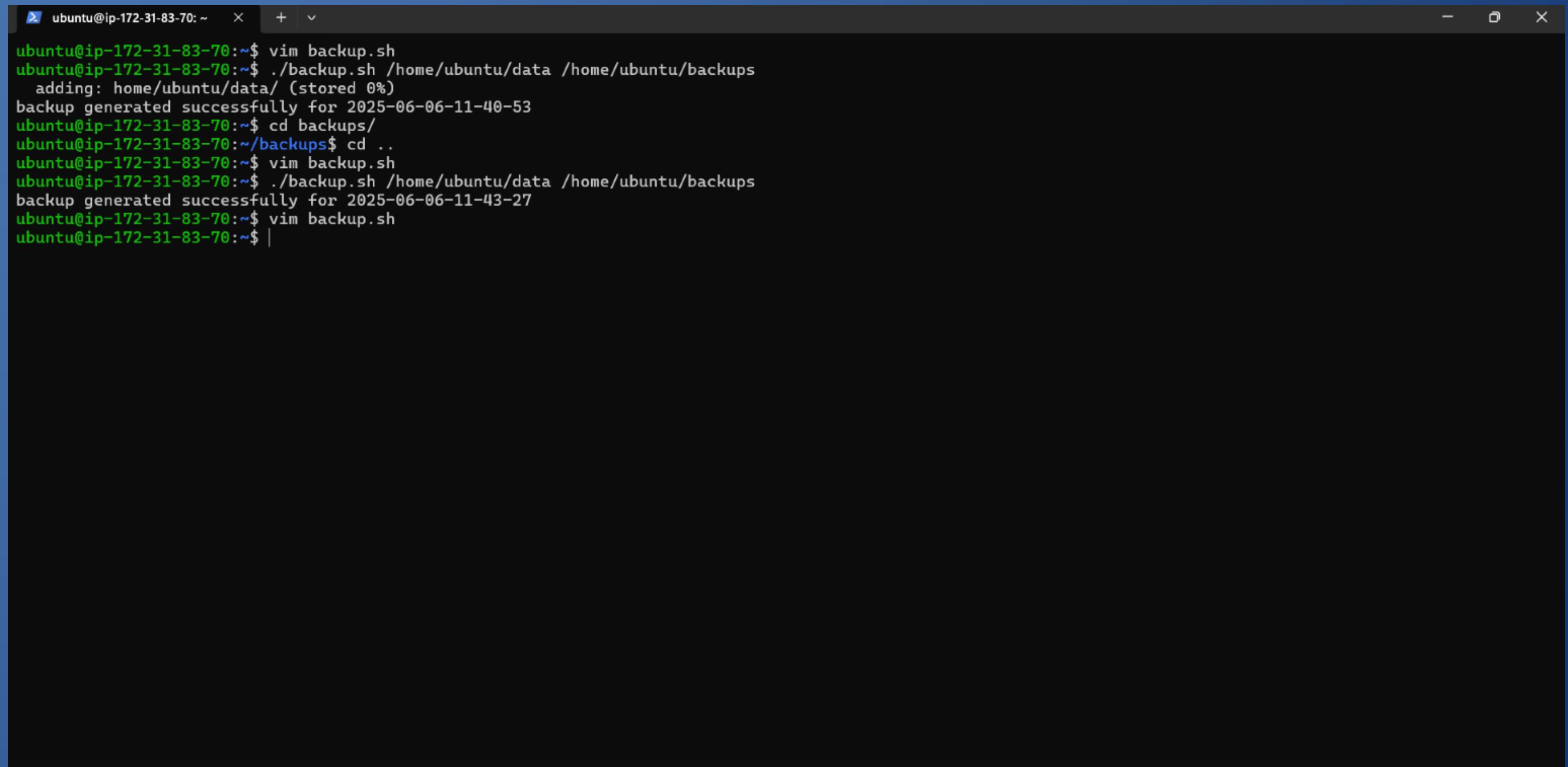
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}"

    if [ $? -eq 0 ]; then
        echo "backup generated successfully for ${timestamp}"
    fi
}
create_backup

~
~
~
~
~
-- INSERT --
32,1          All
```

## TESTING BACKUP CREATION

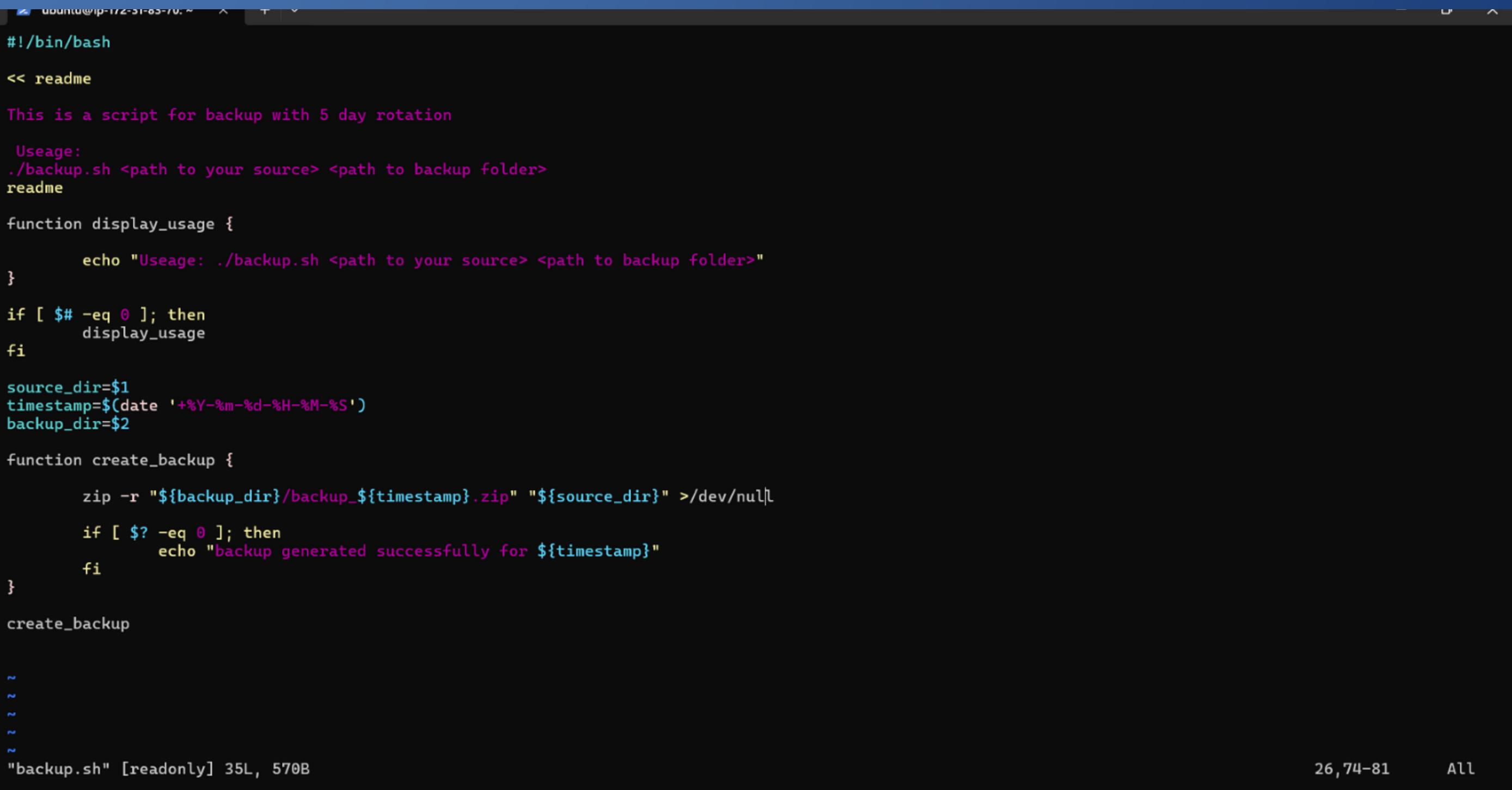
The execution of the backup.sh script with specific source and backup directories: ./backup.sh /home/ubuntu/data /home/ubuntu/backups. The output shows messages indicating successful backup generation, such as "adding: home/ubuntu/data/ (stored 0%)" and "backup generated successfully for [timestamp]". This confirms that the create\_backup function is working as intended, and a backup file is being produced.



```
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
  adding: home/ubuntu/data/ (stored 0%)
backup generated successfully for 2025-06-06-11-40-53
ubuntu@ip-172-31-83-70:~$ cd backups/
ubuntu@ip-172-31-83-70:~/backups$ cd ..
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-06-11-43-27
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ |
```

## SUPPRESSING VERBOSE OUTPUT

This shows a modification to the `create_backup` function within `backup.sh`. The `zip` command is updated to redirect its output to `/dev/null (>/dev/null)`. This change prevents the verbose output of the `zip` command from being displayed on the console during script execution, leading to cleaner and less cluttered output.



The screenshot shows a terminal window with the following content:

```
#!/bin/bash

<< readme
This is a script for backup with 5 day rotation

Usage:
./backup.sh <path to your source> <path to backup folder>
readme

function display_usage {
    echo "Usage: ./backup.sh <path to your source> <path to backup folder>"
}

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null
    if [ $? -eq 0 ]; then
        echo "backup generated successfully for ${timestamp}"
    fi
}

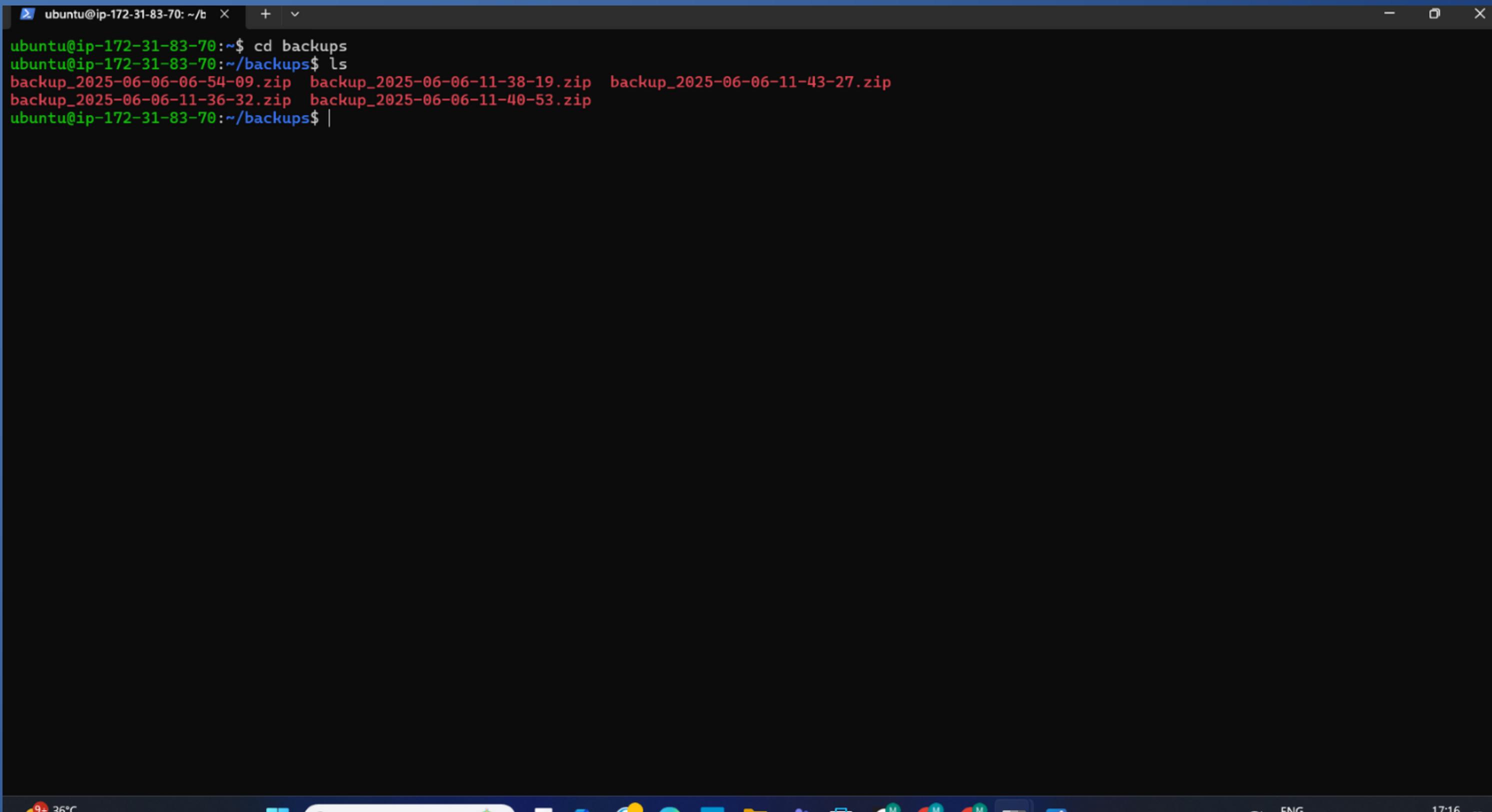
create_backup

~
```

The terminal window title is "ubuntu@ip-172-31-85-70: ~". The status bar at the bottom right shows "26,74-81 All".

## VERIFYING BACKUP FILES

This page shows the user navigating into the backups directory using the cd backups command. Following this, the ls command is used to list the contents of the directory, which displays a series of backup\_YYYY-MM-DD-HH-MM-SS.zip files. This step visually confirms that timestamped backup archives are being successfully created and stored in the designated location.



A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70: ~/b". The terminal displays the following command-line session:

```
ubuntu@ip-172-31-83-70:~$ cd backups
ubuntu@ip-172-31-83-70:~/backups$ ls
backup_2025-06-06-54-09.zip  backup_2025-06-06-11-38-19.zip  backup_2025-06-06-11-43-27.zip
backup_2025-06-06-11-36-32.zip  backup_2025-06-06-11-40-53.zip
ubuntu@ip-172-31-83-70:~/backups$ |
```

The terminal window has a dark blue background and a light blue header bar. The bottom of the screen shows a taskbar with various icons and a system status bar indicating the date and time (17:16), battery level (9+ 36°C), and network connection (FNG).

# VERIFYING BACKUP FILES

This page continues the verification of backup files. It shows the use of the ls -t command within the backups directory. This command lists the backup files sorted by their modification time, with the newest files appearing first. This helps confirm the chronological order and recency of the created archives.

```
ubuntu@ip-172-31-83-70:~$ cd backups/
ubuntu@ip-172-31-83-70:~/backups$ ls
backup_2025-06-06-54-09.zip  backup_2025-06-06-11-38-19.zip  backup_2025-06-06-11-43-27.zip  backup_2025-06-06-11-47-42.zip
backup_2025-06-06-11-36-32.zip  backup_2025-06-06-11-40-53.zip  backup_2025-06-06-11-47-34.zip  backup_2025-06-06-11-47-45.zip
ubuntu@ip-172-31-83-70:~/backups$ ls -t
backup_2025-06-06-11-47-45.zip  backup_2025-06-06-11-47-34.zip  backup_2025-06-06-11-40-53.zip  backup_2025-06-06-11-36-32.zip
backup_2025-06-06-11-47-42.zip  backup_2025-06-06-11-43-27.zip  backup_2025-06-06-11-38-19.zip  backup_2025-06-06-06-54-09.zip
ubuntu@ip-172-31-83-70:~/backups$ cd ..
ubuntu@ip-172-31-83-70:~$ vim backup.sh
```

# BEGINNING BACKUP ROTATION FUNCTION

This marks the introduction of the perform\_rotation function within the backup.sh script. This function is designed to manage backup retention. It begins by listing backup files sorted by modification time (`ls -t "${backup_dir}/backup.zip"`) and stores them in an array named backups. The content of this backups array is then echoed, likely for debugging or initial visual verification.

```
ubuntu@ip-172-31-83-70: ~      + | v
./backup.sh <path to your source> <path to backup folder>
readme

function display_usage {

    echo "Usage: ./backup.sh <path to your source> <path to backup folder>"

}

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {

    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null

    if [ $? -eq 0 ]; then
        echo "backup generated successfully for ${timestamp}"
    fi
}

function perform_rotation {
    backups=($(ls -t "${backup_dir}/backup_*.zip"))
    echo "${backups[@]}"
}

create_backup
perform_rotation

~
~
~
~
~
:wq!|
```

# DISPLAYING ALL BACKUPS (VISUAL TEST)

This illustrates a visual test of the `perform_rotation` function. After the script is executed, the output shows a long list of all existing backup files, similar to the output from `ls -t`. This demonstrates the functionality of the `echo "${backups[@]}"` command within the `perform_rotation` function, confirming that the array is correctly populated and its contents can be displayed.

```
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-06-12-45-46
/home/ubuntu/backups/backup_2025-06-06-12-45-46.zip /home/ubuntu/backups/backup_2025-06-12-24-00.zip /home/ubuntu/backups/backup_2025-06-12-45-41.zip /home/ubuntu/backups/backup_2025-06-11-47-45.zip /home/ubuntu/backups/backup_2025-06-11-47-42.zip /home/ubuntu/backups/backup_2025-06-11-47-34.zip /home/ubuntu/backups/backup_2025-06-11-43-27.zip /home/ubuntu/backups/backup_2025-06-11-40-53.zip /home/ubuntu/backups/backup_2025-06-11-38-19.zip /home/ubuntu/backups/backup_2025-06-11-36-32.zip /home/ubuntu/backups/backup_2025-06-06-54-09.zip
ubuntu@ip-172-31-83-70:~$ |
```

# REFINING ROTATION CONDITION

This shows a further refinement to the perform\_rotation function within backup.sh. Specifically, an if condition if ["\${#backups[@]}" -gt 5 ] is introduced. This condition checks if the total number of backup files listed in the backups array is greater than 5. This is a crucial step towards implementing the 5-day rotation policy, as it determines when older backups need to be removed.

```
function display_usage {
    echo "Usage: ./backup.sh <path to your source> <path to backup folder>"
}

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null
    if [ $? -eq 0 ]; then
        echo "Backup generated successfully for ${timestamp}"
    fi
}

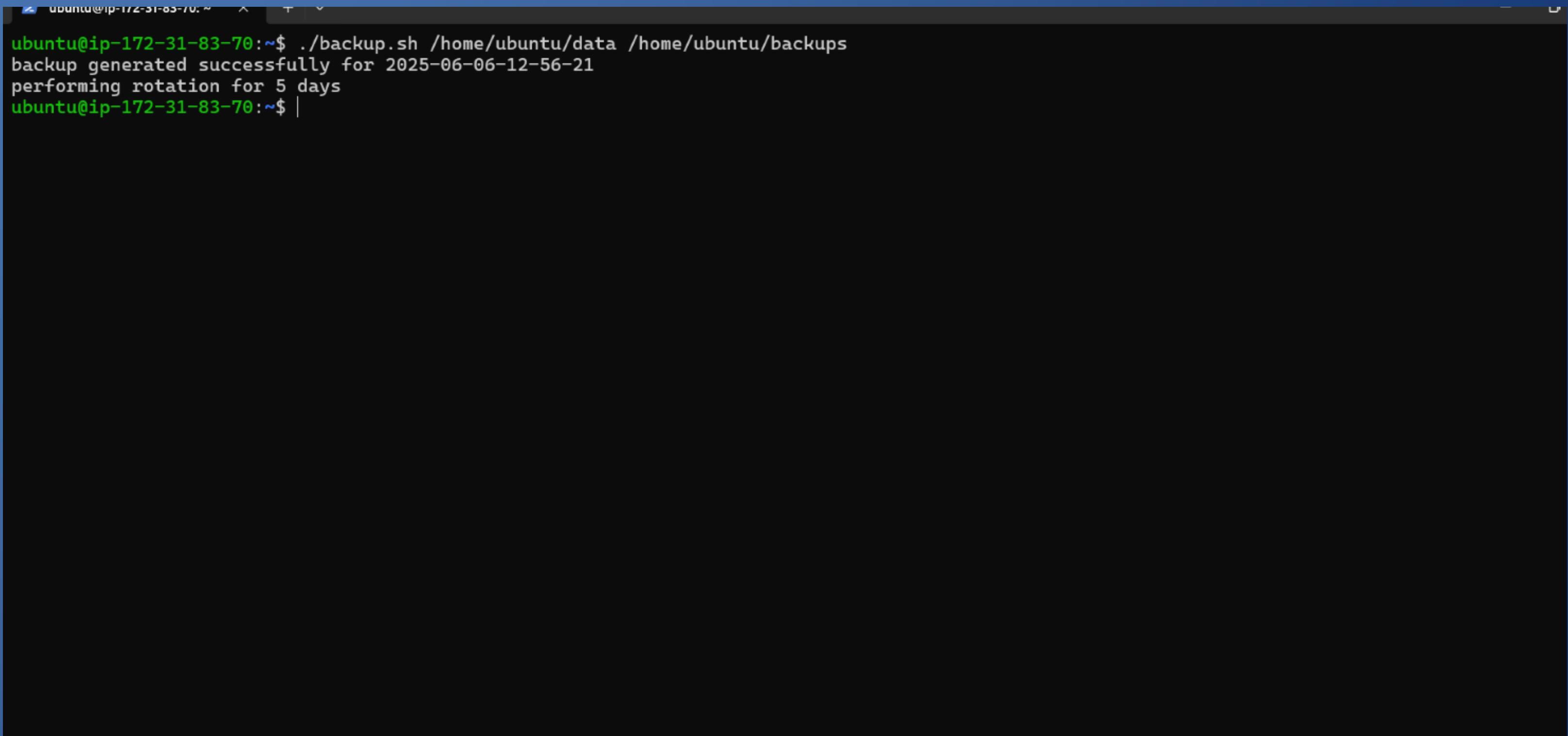
function perform_rotation {
    backups=$(ls -t "${backup_dir}/backup_*.zip" 2>/dev/null)

    if [ "${#backups[@]}" -gt 5 ]; then
        echo "Performing rotation for 5 days"
    fi
}

create_backup
perform_rotation
~:wq!
```

## ROTATION MESSAGE

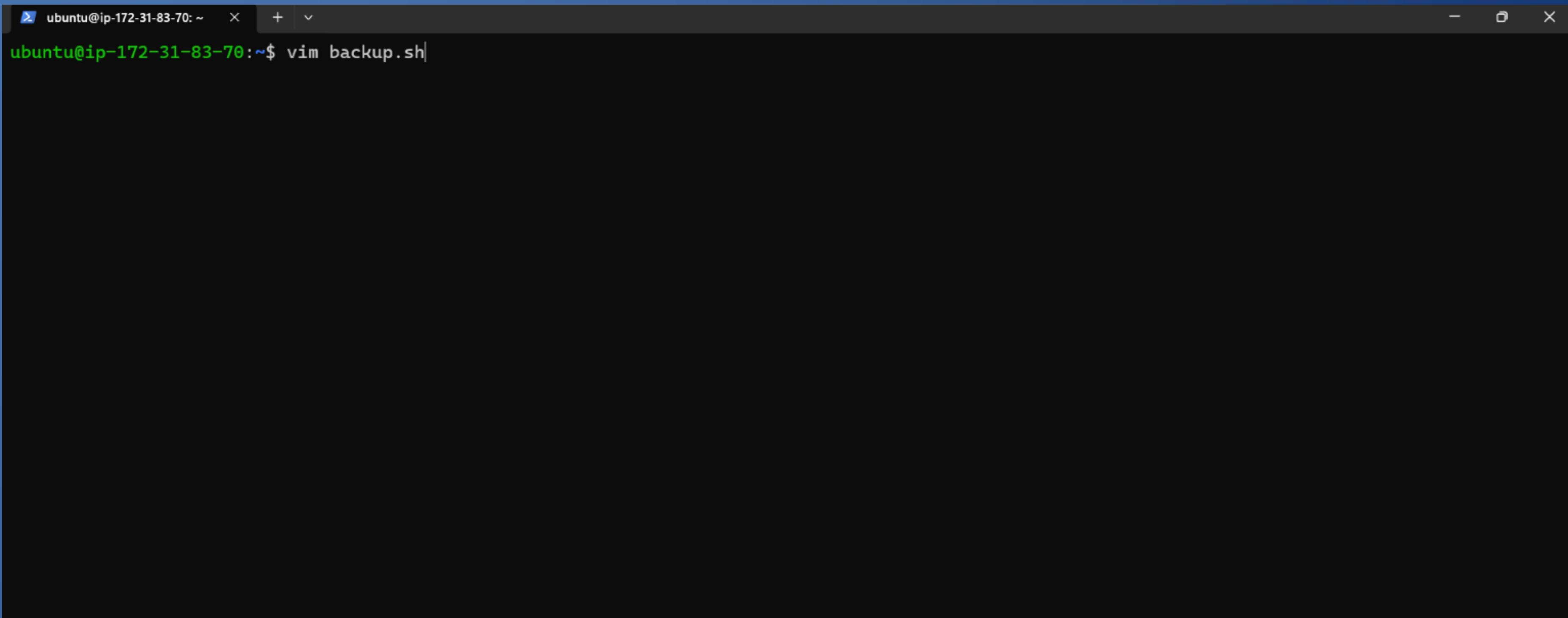
This shows the backup.sh script now printing the message "performing rotation for 5 days" during execution. This output appears when the condition for backup rotation is met, providing a clear indication that the script is actively managing older backups according to the defined 5-day policy.

A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70:~". The window contains the following text:

```
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-06-12-56-21
performing rotation for 5 days
ubuntu@ip-172-31-83-70:~$ |
```

The text is in white on a black background, with the terminal prompt and command in green.

## ENTERING THE VIM BACKUP.SH COMMAND



A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70: ~". The window shows the command "vim backup.sh" being typed at the prompt. The terminal has a dark background with light-colored text. The title bar is visible at the top.

```
ubuntu@ip-172-31-83-70:~$ vim backup.sh
```

## IDENTIFYING BACKUPS FOR REMOVAL

This page details a crucial step in the `perform_rotation` function: identifying which backups should be removed. The code `backups_to_remove=(${backups[@]:5})` is shown, which selects all elements in the `backups` array starting from the sixth element (index 5), effectively targeting backups older than the five most recent. The line `echo "${backups_to_remove[@]}"` is also present, likely for displaying these identified backups before deletion.

```
ubuntu@ip-172-31-83-70: ~ + ^X - X
echo "Usage: ./backup.sh <path to your source> <path to backup folder>"}
if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null
    if [ $? -eq 0 ]; then
        echo "Backup generated successfully for ${timestamp}"
    fi
}

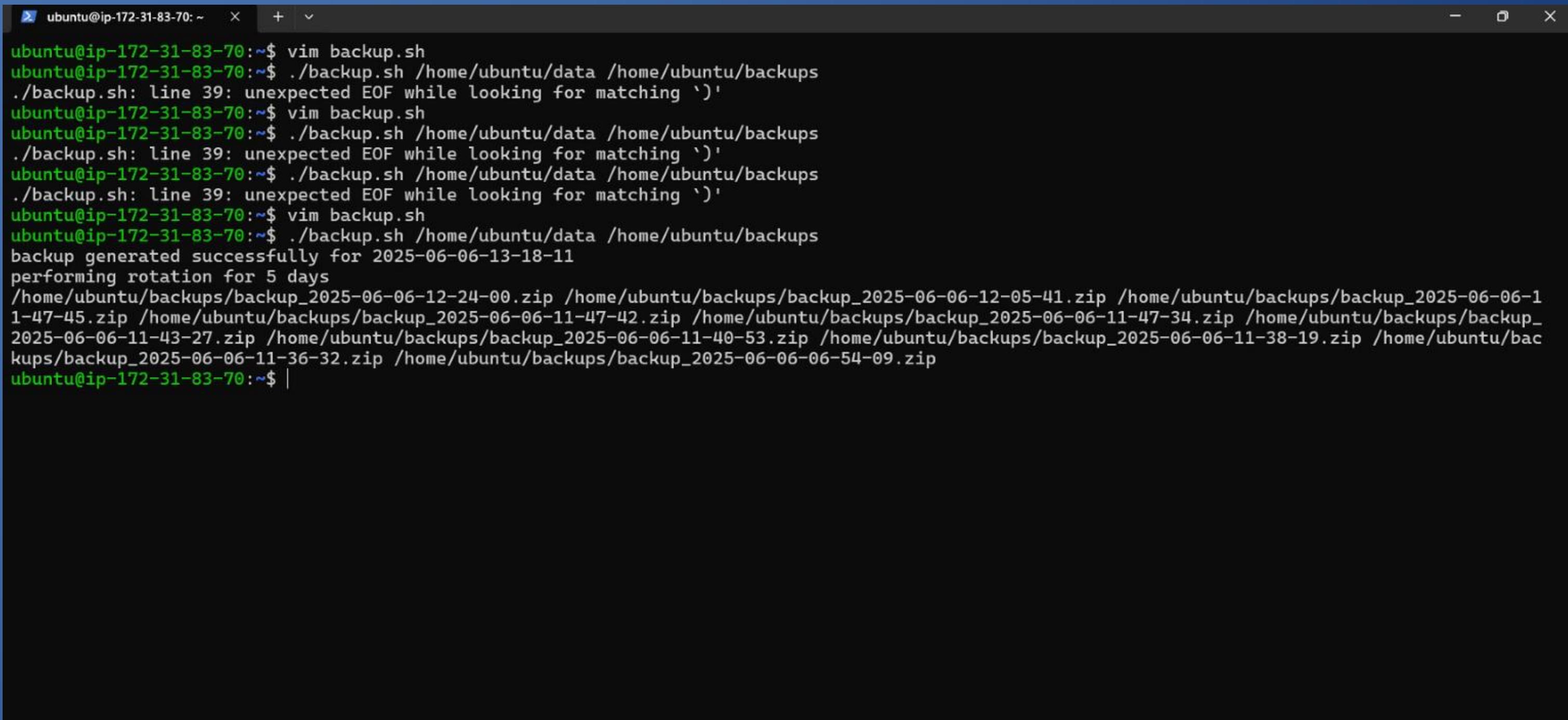
function perform_rotation {
    backups=($(ls -t "${backup_dir}/backup_*.zip" 2>/dev/null))

    if [ "${#backups[@]}" -gt 5 ]; then
        echo "Performing rotation for 5 days"
        backups_to_remove=("${backups[@]:5}")
        echo "${backups_to_remove[@]}"
    fi
}

create_backup
perform_rotation
-- INSERT --
```

# SCRIPT SYNTAX ERROR (INTERIM STATE)

This page shows an error message encountered during script execution: "line 39: unexpected EOF while looking for matching ')'. This indicates a syntax issue or an incomplete code block within the backup.sh script, specifically on line 39. This suggests the script was in an interim state during development, requiring debugging.

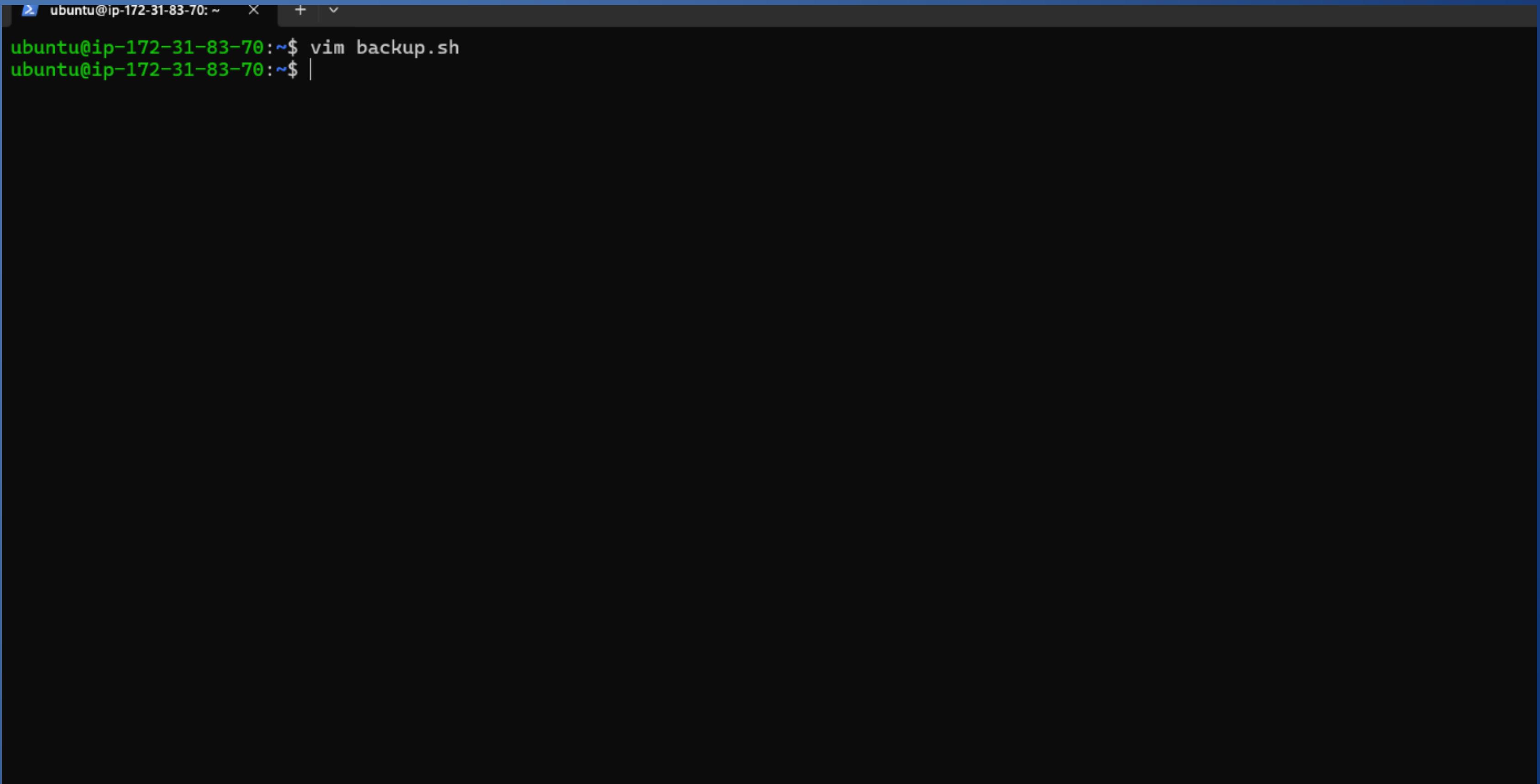


The screenshot shows a terminal window with a dark background and light-colored text. The terminal title is "ubuntu@ip-172-31-83-70:~". The session starts with the user running "vim backup.sh" to view the script. They then execute "./backup.sh /home/ubuntu/data /home/ubuntu/backups", which triggers the syntax error: "./backup.sh: line 39: unexpected EOF while looking for matching ')'. The user repeats this process of viewing the script and running the command several times, each time encountering the same error. Finally, they run the command successfully, generating a backup for the date 2025-06-06-18-11, performing a rotation for 5 days, and listing the resulting zip files in the backups directory.

```
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
./backup.sh: line 39: unexpected EOF while looking for matching ')'
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
./backup.sh: line 39: unexpected EOF while looking for matching ')'
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
./backup.sh: line 39: unexpected EOF while looking for matching ')'
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-06-18-11
performing rotation for 5 days
/home/ubuntu/backups/backup_2025-06-12-24-00.zip /home/ubuntu/backups/backup_2025-06-06-41.zip /home/ubuntu/backups/backup_2025-06-06-1
1-47-45.zip /home/ubuntu/backups/backup_2025-06-06-11-47-42.zip /home/ubuntu/backups/backup_2025-06-06-11-47-34.zip /home/ubuntu/backups/backup_
2025-06-06-11-43-27.zip /home/ubuntu/backups/backup_2025-06-06-11-40-53.zip /home/ubuntu/backups/backup_2025-06-06-11-38-19.zip /home/ubuntu/bac
kups/backup_2025-06-06-11-36-32.zip /home/ubuntu/backups/backup_2025-06-06-09.zip
ubuntu@ip-172-31-83-70:~$ |
```

# ENTERING THE COMMAND VIM

## BACKUP.SH



A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70: ~". The window shows the command "vim backup.sh" being typed at the prompt. The text "ubuntu@ip-172-31-83-70:~\$ vim backup.sh" is displayed in green, while the cursor is shown as a vertical bar at the end of the line.

```
ubuntu@ip-172-31-83-70:~$ vim backup.sh
```

# IMPLEMENTING BACKUP REMOVAL

This page details the crucial step of implementing the actual deletion of old backup files within the perform\_rotation function. A for loop is introduced: for backup in "\${backups\_to\_remove[@]}"; do rm -f "\$backup"; done. This loop iterates through the backups\_to\_remove array (which contains backups older than the 5 most recent) and uses rm -f to forcibly remove each specified backup file.

```
ubuntu@ip-172-31-83-70: ~  +  -  X
echo "Usage: ./backup.sh <path to your source> <path to backup folder>"
}

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null
    if [ $? -eq 0 ]; then
        echo "backup generated successfully for ${timestamp}"
    fi
}

function perform_rotation {
    backups=($(ls -t "${backup_dir}/backup_*.zip" 2>/dev/null))

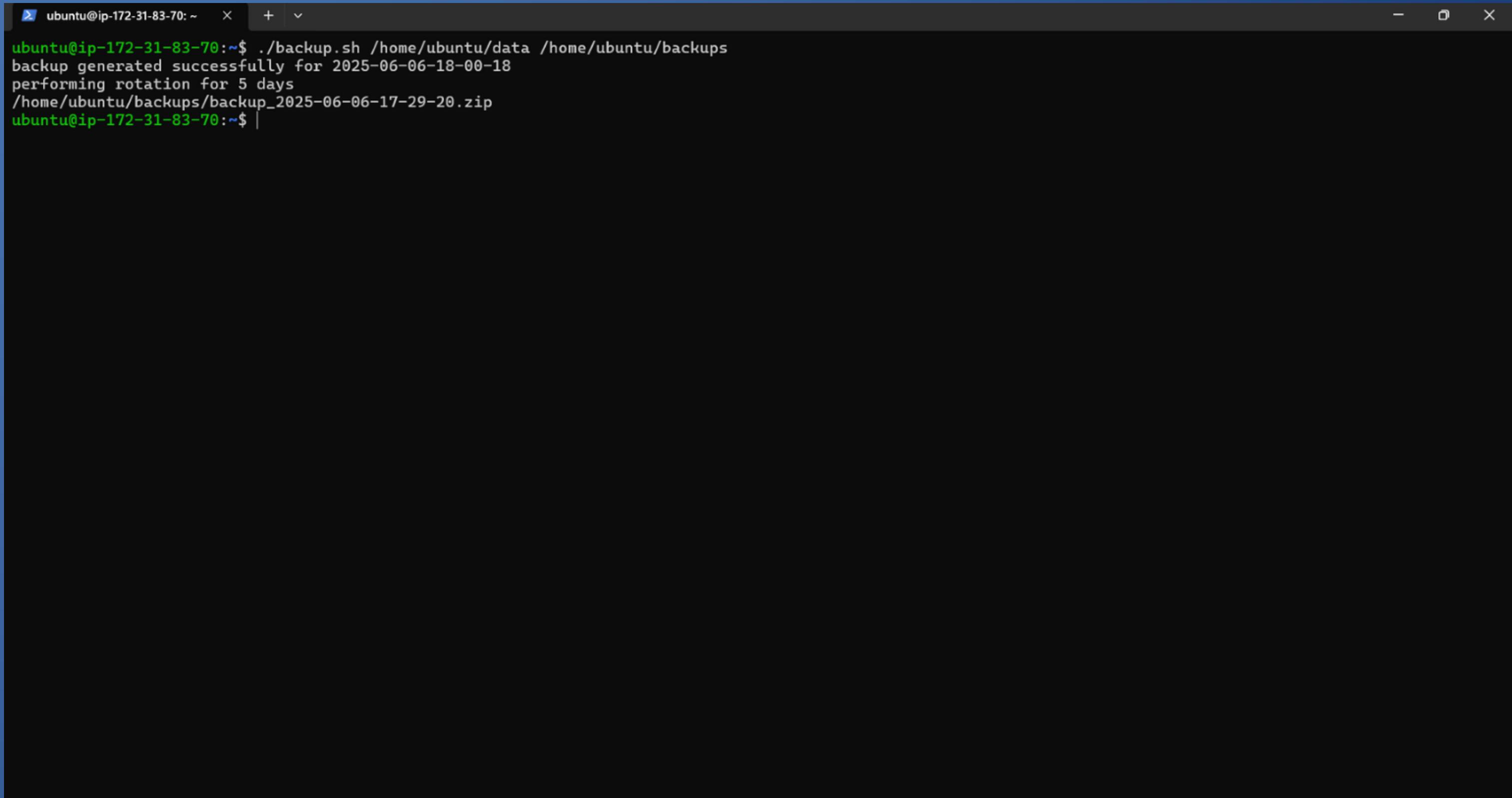
    if [ "${#backups[@]}" -gt 5 ]; then
        echo "performing rotation for 5 days"
        backups_to_remove=("${backups[@]:5}")
        echo "${backups_to_remove[@]}"

        for backup in "${backups_to_remove[@]}";
        do
            rm -f $[backup]
        done
    fi
}

:wq!
```

# TESTING BACKUP REMOVAL

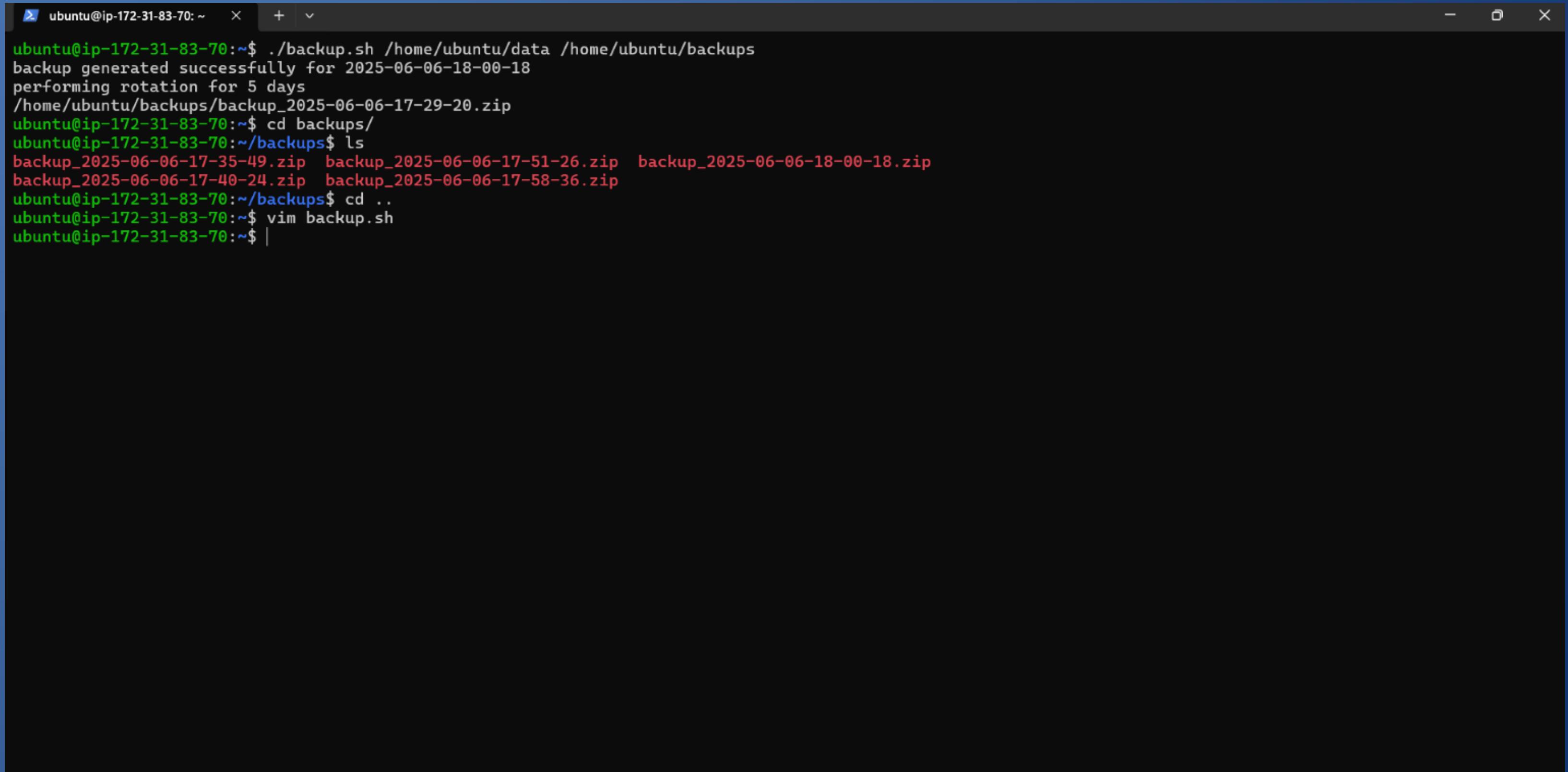
This shows the initial results of testing the implemented backup removal logic. After running the backup.sh script, the output confirms "backup generated successfully..." and also indicates "performing rotation for 5 days." This signifies that both the backup creation and the rotation processes are being triggered. This page visually implies the start of the verification that older backups are indeed being managed.



```
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-06-18-00-18
performing rotation for 5 days
/home/ubuntu/backups/backup_2025-06-06-17-29-20.zip
ubuntu@ip-172-31-83-70:~$ |
```

## CONFIRMING BACKUP REMOVAL

This page continues the verification of the backup rotation. After the script runs, listing the backups (e.g., with ls) shows fewer files than before, specifically, only the five most recent backups remain. This visual confirmation indicates that the perform\_rotation function, including the rm -f command for deletion, has successfully removed the older backup files as intended by the 5-day rotation policy.



The screenshot shows a terminal window with the following session:

```
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-06-00-18
performing rotation for 5 days
/home/ubuntu/backups/backup_2025-06-06-17-29-20.zip
ubuntu@ip-172-31-83-70:~$ cd backups/
ubuntu@ip-172-31-83-70:~/backups$ ls
backup_2025-06-06-17-35-49.zip  backup_2025-06-06-17-51-26.zip  backup_2025-06-06-18-00-18.zip
backup_2025-06-06-17-40-24.zip  backup_2025-06-06-17-58-36.zip
ubuntu@ip-172-31-83-70:~/backups$ cd ..
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ |
```

# CONTINUED VERIFICATION

Given the sequential nature of the document, it contains further visual verification of the backup.sh script's execution. This would include confirming the output and behavior of the script after the backup rotation logic was implemented and tested, ensuring everything worked as expected before proceeding to automation.

```
ubuntu@ip-172-31-83-70: ~ + - X
if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

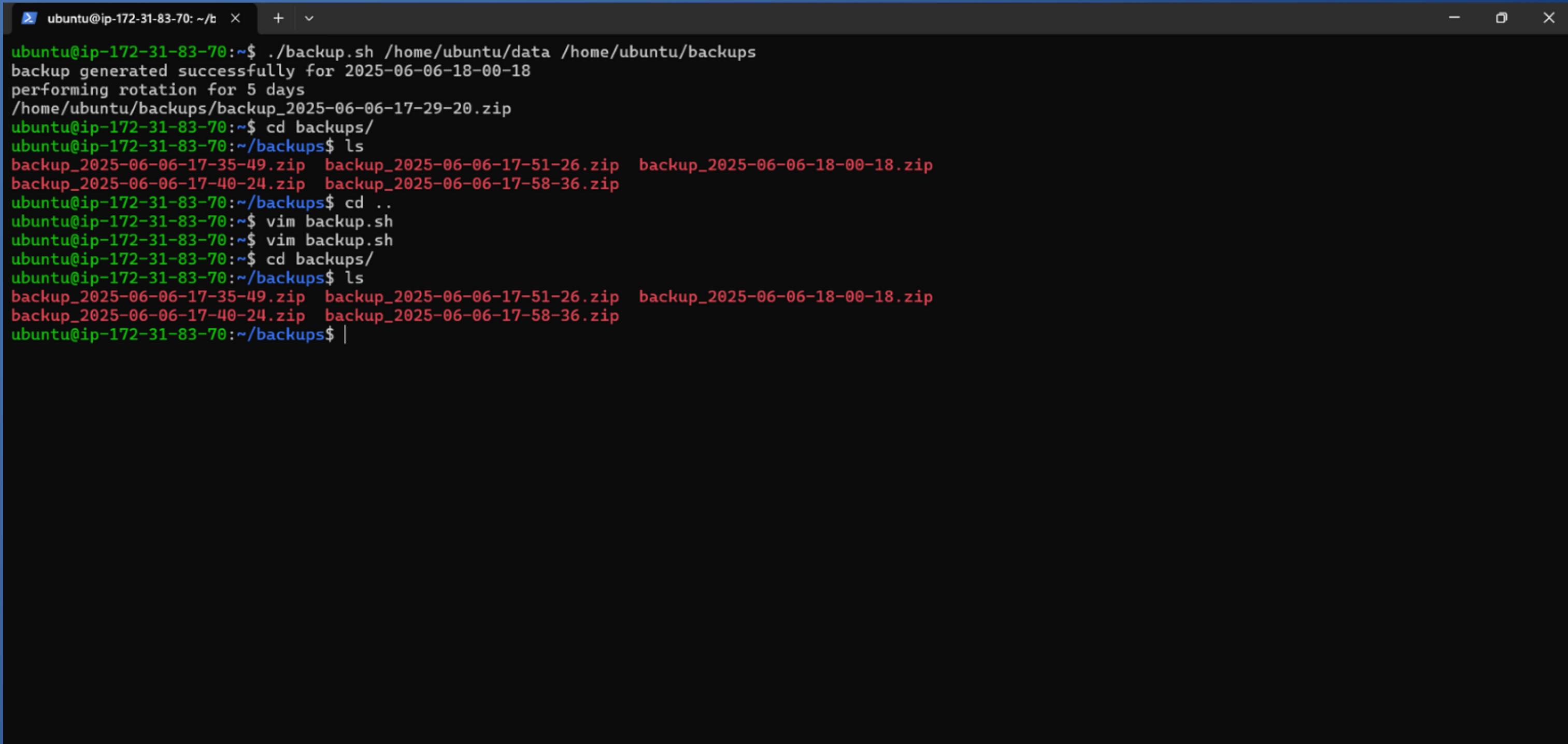
function create_backup {
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null
    if [ $? -eq 0 ]; then
        echo "backup generated successfully for ${timestamp}"
    fi
}

function perform_rotation {
    backups=$(ls -t "${backup_dir}/backup_*.zip" 2>/dev/null)
    if [ "${#backups[@]}" -gt 5 ]; then
        echo "performing rotation for 5 days"
        backups_to_remove="${backups[@]:5}"
        for backup in "${backups_to_remove[@]}";
        do
            rm -f ${backup}
        done
    fi
}

create_backup
perform_rotation
-- INSERT --
```

## TRANSITION TO AUTOMATION

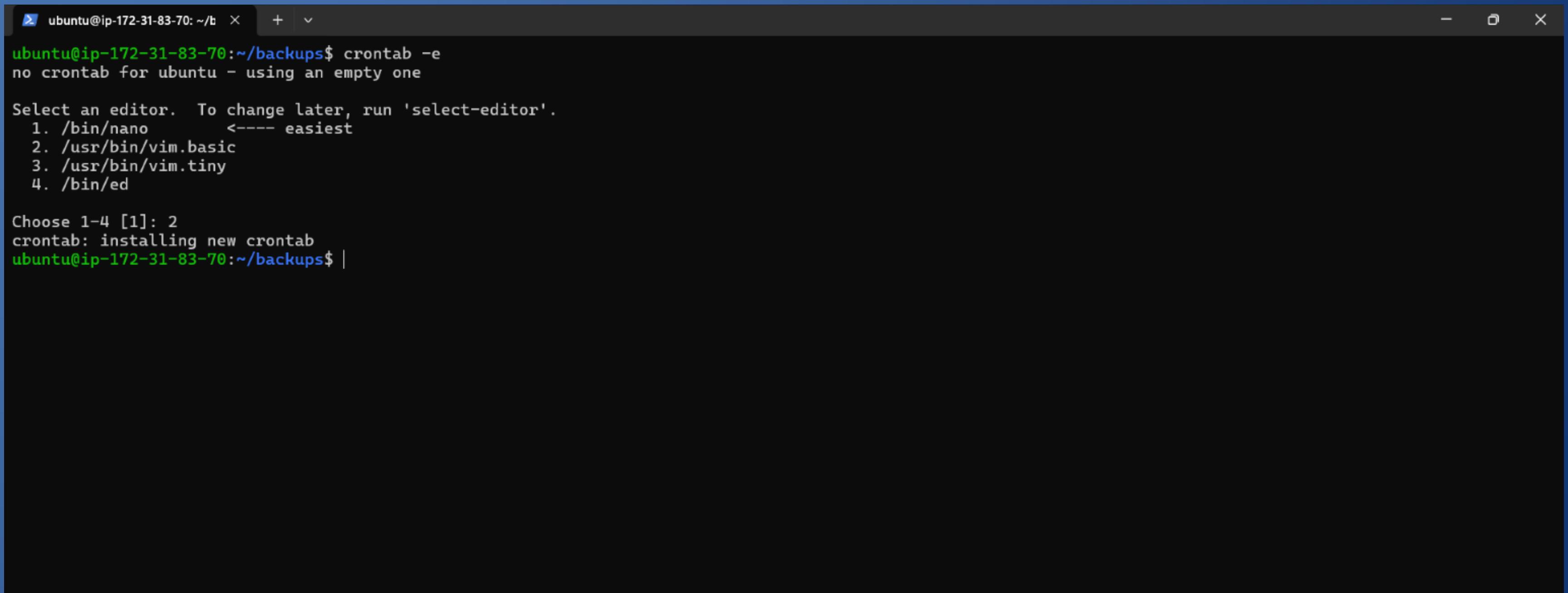
IT serves as a transitional point in the documentation. Given the preceding sections on script development and testing, This contains further visual verification of the backup.sh script's output or its behavior after the rotation logic has been fully implemented. It bridges the gap between script functionality confirmation and the final automation setup.



```
ubuntu@ip-172-31-83-70:~/b ~ + ^ - X
ubuntu@ip-172-31-83-70:~$ ./backup.sh /home/ubuntu/data /home/ubuntu/backups
backup generated successfully for 2025-06-18-00-18
performing rotation for 5 days
/home/ubuntu/backups/backup_2025-06-17-29-20.zip
ubuntu@ip-172-31-83-70:~$ cd backups/
ubuntu@ip-172-31-83-70:~/backups$ ls
backup_2025-06-17-35-49.zip backup_2025-06-17-51-26.zip backup_2025-06-18-00-18.zip
backup_2025-06-17-40-24.zip backup_2025-06-17-58-36.zip
ubuntu@ip-172-31-83-70:~/backups$ cd ..
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ vim backup.sh
ubuntu@ip-172-31-83-70:~$ cd backups/
ubuntu@ip-172-31-83-70:~/backups$ ls
backup_2025-06-17-35-49.zip backup_2025-06-17-51-26.zip backup_2025-06-18-00-18.zip
backup_2025-06-17-40-24.zip backup_2025-06-17-58-36.zip
ubuntu@ip-172-31-83-70:~/backups$ |
```

## INITIATING CRON JOB SETUP

This page shows the command used to begin the process of automating the backup script. The user executes crontab -e, which opens the cron table (a time-based job scheduler) for editing. This is the first step towards setting up backup.sh to run automatically at defined intervals without manual intervention.



A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70: ~/b". The window displays the following text:

```
ubuntu@ip-172-31-83-70:~/backups$ crontab -e
no crontab for ubuntu - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed

Choose 1-4 [1]: 2
crontab: installing new crontab
ubuntu@ip-172-31-83-70:~/backups$ |
```

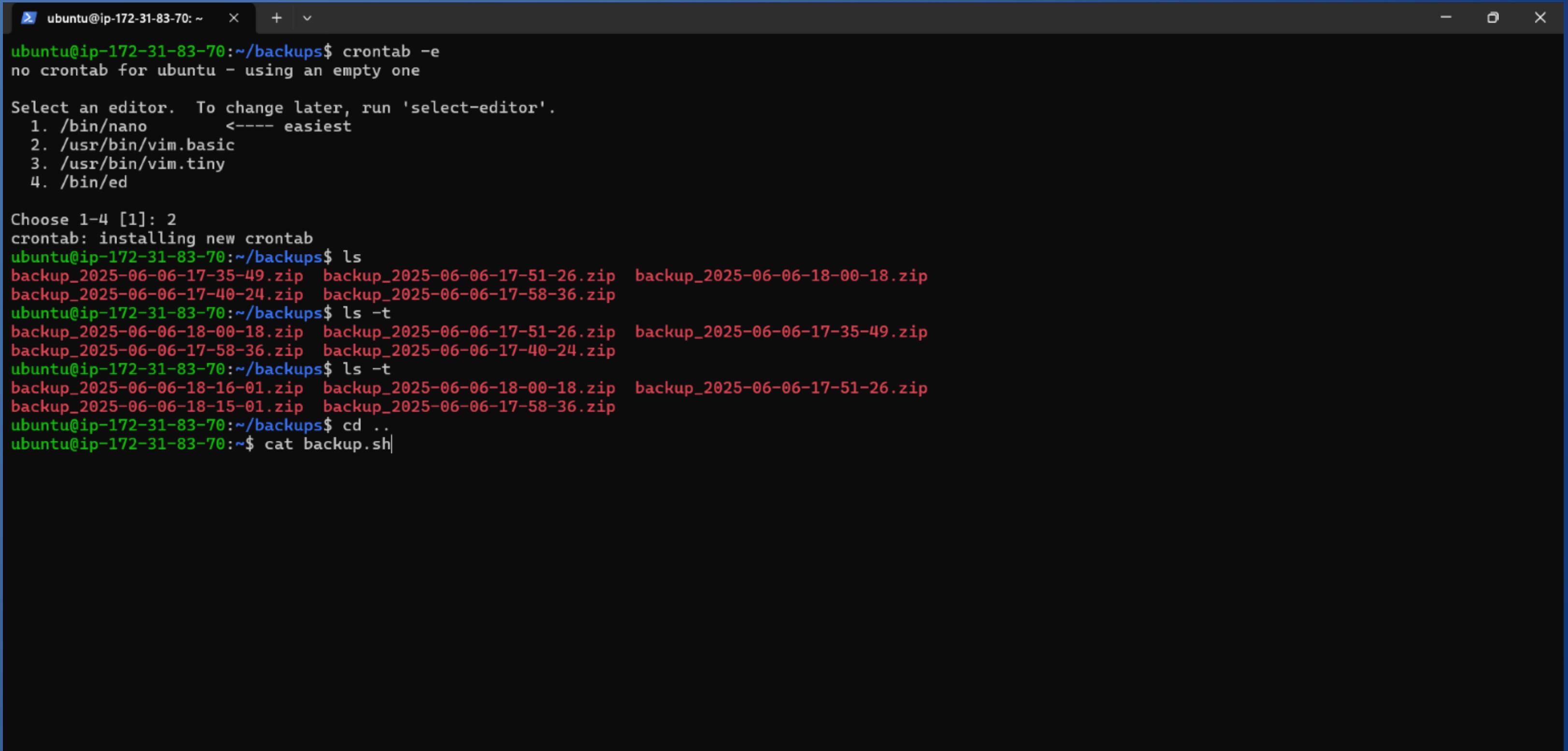
# ADDING CRON JOB FOR AUTOMATION

This page details the crucial step of automating the backup.sh script using cron. The crontab -e command is shown, which opens the user's cron table for editing. Within this editor, a specific cron entry is added: \* \* \* \* \* bash /home/ubuntu/backup.sh /home/ubuntu/data /home/ubuntu/backups. This line instructs the system to execute the backup.sh script every minute, ensuring regular, automated backups are performed.

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * bash /home/ubuntu/backup.sh /home/ubuntu/data /home/ubuntu/backups
```

## CRONTAB SETUP CONFIRMATION & BACKUP LISTING

This page confirms the successful setup of the automated backup schedule. The system displays "crontab: installing new crontab" after the user has configured the cron job. Following this, the user verifies the presence of existing backups by using the ls command within the backups directory, which lists various timestamped .zip archives. The command cat backup.sh is also shown, indicating the user might be reviewing the final content of their backup script.



The screenshot shows a terminal window with the following session:

```
ubuntu@ip-172-31-83-70:~$ crontab -e
no crontab for ubuntu - using an empty one

Select an editor. To change later, run 'select-editor'.
1. /bin/nano      <---- easiest
2. /usr/bin/vim.basic
3. /usr/bin/vim.tiny
4. /bin/ed

Choose 1-4 [1]: 2
crontab: installing new crontab
ubuntu@ip-172-31-83-70:~/backups$ ls
backup_2025-06-06-17-35-49.zip  backup_2025-06-06-17-51-26.zip  backup_2025-06-06-18-00-18.zip
backup_2025-06-06-17-40-24.zip  backup_2025-06-06-17-58-36.zip
ubuntu@ip-172-31-83-70:~/backups$ ls -t
backup_2025-06-06-18-00-18.zip  backup_2025-06-06-17-51-26.zip  backup_2025-06-06-17-35-49.zip
backup_2025-06-06-17-58-36.zip  backup_2025-06-06-17-40-24.zip
ubuntu@ip-172-31-83-70:~/backups$ ls -t
backup_2025-06-06-18-16-01.zip  backup_2025-06-06-18-00-18.zip  backup_2025-06-06-17-51-26.zip
backup_2025-06-06-18-15-01.zip  backup_2025-06-06-17-58-36.zip
ubuntu@ip-172-31-83-70:~/backups$ cd ..
ubuntu@ip-172-31-83-70:~$ cat backup.sh
```

# AUTOMATED BACKUP SCRIPT OVERVIEW

This document presents the full backup.sh script. It includes functions for displaying proper usage, creating timestamped zip backups with error handling and suppressed output, and performing rotation. The perform\_rotation function is key; it lists backups, identifies those older than the five most recent, and deletes them. The script concludes by running both the backup creation and rotation processes.

```
ubuntu@ip-172-31-83-70: ~      + | - X

if [ $# -eq 0 ]; then
    display_usage
fi

source_dir=$1
timestamp=$(date '+%Y-%m-%d-%H-%M-%S')
backup_dir=$2

function create_backup {
    zip -r "${backup_dir}/backup_${timestamp}.zip" "${source_dir}" >/dev/null
    if [ $? -eq 0 ]; then
        echo "backup generated successfully for ${timestamp}"
    fi
}

function perform_rotation {
    backups=($(ls -t "${backup_dir}/backup_*.zip" 2>/dev/null))

    if [ "${#backups[@]}" -gt 5 ]; then
        echo "performing rotation for 5 days"
        backups_to_remove="${backups[@]:5}"

        for backup in "${backups_to_remove[@]}";
        do
            rm -f $backup
        done
    fi
}

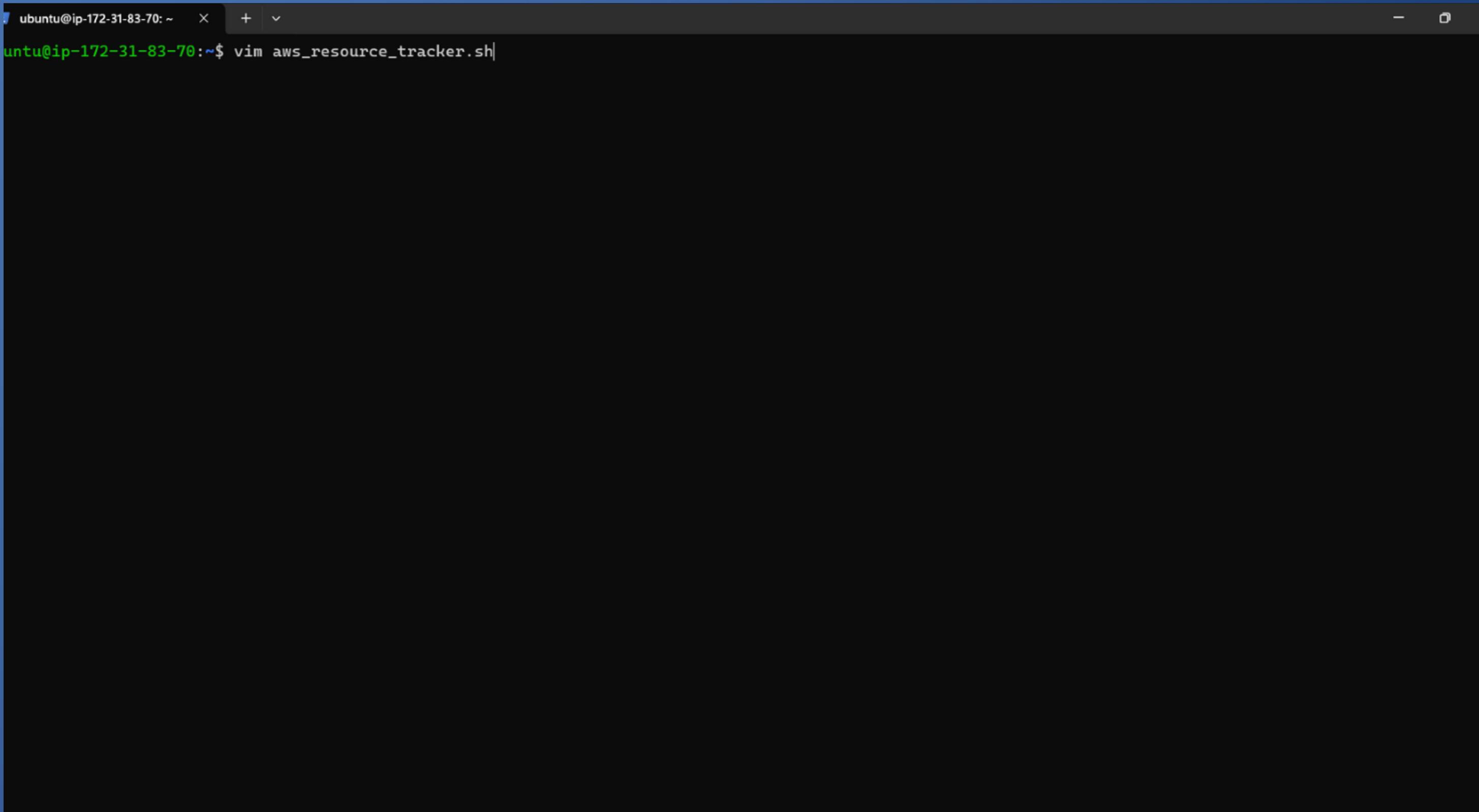
create_backup
perform_rotation
```

# AWS RESOURCE TRACKER

This project centers on an "AWS Resource Tracker". I created and refined a bash script, `aws_resource_tracker.sh`, to report on AWS S3, EC2, Lambda, and IAM user usage. Key steps included configuring AWS access, making the script executable, running it to view resource output, and using jq to filter specific EC2 instance IDs. The script was further modified to include debugging with `set -x` and to redirect or append output to a file named `resourceTracker`.

## INITIAL SETUP & SCRIPT CREATION

This page shows a that i logged into an Ubuntu terminal, initiating the creation of a shell script named aws\_resource\_tracker.sh using the vim editor.

A screenshot of a terminal window titled "ubuntu@ip-172-31-83-70: ~". The window shows the command "vim aws\_resource\_tracker.sh" being typed at the prompt. The terminal has a dark background with white text and a black header bar.

```
ubuntu@ip-172-31-83-70:~$ vim aws_resource_tracker.sh
```

# SCRIPT CONTENT - RESOURCE TRACKING

This page displays the initial content of the `aws_resource_tracker.sh` script. It's a bash script authored by Wasim on June 3rd, version v1, designed to report AWS resource usage. The script includes commands to list S3 buckets (`aws s3 ls`), describe EC2 instances (`aws ec2 describe-instances`), list Lambda functions (`aws lambda list-functions`), and list IAM users (`aws iam list-users`).

# AWS CONFIGURATION & SCRIPT EXECUTION OUTPUT

This page captures the configuration of AWS access keys, secret access keys, default region (us-east-1), and output format (json). Following configuration, the aws\_resource\_tracker.sh script is made executable with chmod 777 and then executed. The output shows detailed JSON for EC2 instances, including a ReservationId, OwnerId, VolumeId ("vol-093be5233381bffffd"), and GroupId ("sg-0ed000d3c8012dd57").

```
ubuntu@ip-172-31-83-70: ~ x + v
AWS Access Key ID [*****PNHG]: AKIA4Q3QK7TNOV3PNHG
AWS Secret Access Key [*****qiIW]: Vq7ZOXUunToAAN2i53xoUDNERwN88qMSYzXrqiIW
Default region name [us-east-1]:
Default output format [json]:
ubuntu@ip-172-31-83-70:~$ vim aws_resource_tracker.sh
ubuntu@ip-172-31-83-70:~$ chmod 777 aws_resource_tracker.sh
ubuntu@ip-172-31-83-70:~$ ./aws_resource_tracker.sh
{
  "Reservations": [
    {
      "ReservationId": "r-092d8befd54a926f7",
      "OwnerId": "860839673074",
      "Groups": [],
      "Instances": [
        {
          "Architecture": "x86_64",
          "BlockDeviceMappings": [
            {
              "DeviceName": "/dev/sda1",
              "Ebs": {
                "AttachTime": "2025-05-27T09:59:01+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-093be5233381bffffd"
              }
            }
          ],
          "ClientToken": "77a097a2-909c-4e1f-a856-8df426a3372f",
          "EbsOptimized": false,
          "EnaSupport": true,
          "Hypervisor": "xen",
          "NetworkInterfaces": [
            {
              "Attachment": {
                "AttachTime": "2025-05-27T09:59:00+00:00",
                "AttachmentId": "eni-attach-00fa74c183d6ef613",
                "DeleteOnTermination": true,
                "DeviceIndex": 0,
                "Status": "attached",
                "NetworkCardIndex": 0
              }
            }
          ]
        }
      ]
    }
  ]
}
```

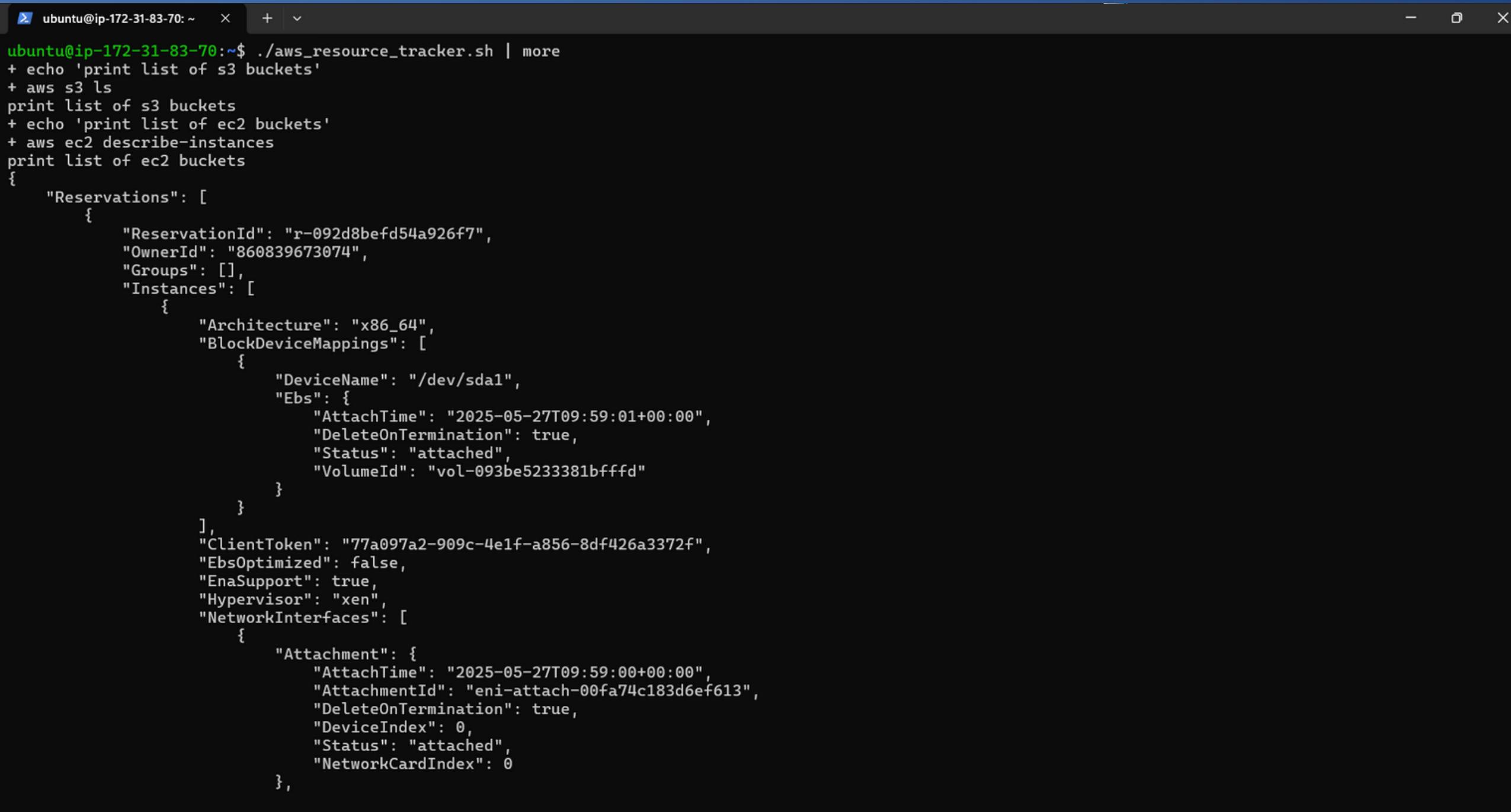
## CONTINUED SCRIPT OUTPUT (EC2, LAMBDA, IAM)

This page continues the output from the aws\_resource\_tracker.sh script. It reiterates details for an EC2 instance, including its Architecture, BlockDeviceMappings, and NetworkInterfaces. It also shows empty arrays for "Functions" (Lambda) and "Users" (IAM), indicating no Lambda functions or IAM users were found at the time of execution.

```
ubuntu@ip-172-31-83-70:~$ ./aws_resource_tracker.sh | jq .Instances[0] | less
{
  "Architecture": "x86_64",
  "BlockDeviceMappings": [
    {
      "DeviceName": "/dev/sda1",
      "Ebs": {
        "AttachTime": "2025-05-27T09:59:01+00:00",
        "DeleteOnTermination": true,
        "Status": "attached",
        "VolumeId": "vol-093be5233381bffffd"
      }
    }
  ],
  "ClientToken": "77a097a2-909c-4e1f-a856-8df426a3372f",
  "EbsOptimized": false,
  "EnaSupport": true,
  "Hypervisor": "xen",
  "NetworkInterfaces": [
    {
      "Attachment": {
        "AttachTime": "2025-05-27T09:59:00+00:00",
        "AttachmentId": "eni-attach-00fa74c183d6ef613",
        "DeleteOnTermination": true,
        "DeviceIndex": 0,
        "Status": "attached",
        "NetworkCardIndex": 0
      },
      "Description": "",
      "Groups": [
        {
          "GroupId": "sg-0ed000d3c8012dd57",
          "GroupName": "launch-wizard-7"
        }
      ]
    }
  ],
  "Functions": [],
  "Users": []
}
ubuntu@ip-172-31-83-70:~$ |
```

# SCRIPT EXECUTION WITH MORE & INITIAL OUTPUT

This page shows the execution of the aws\_resource\_tracker.sh script piped to more. The set -x command reveals the execution flow, printing "print list of s3 buckets" before running aws s3 ls, and "print list of ec2 buckets" before aws ec2 describe-instances. The page then displays the beginning of the detailed JSON output for EC2 instances.



```
ubuntu@ip-172-31-83-70:~$ ./aws_resource_tracker.sh | more
+ echo 'print list of s3 buckets'
+ aws s3 ls
print list of s3 buckets
+ echo 'print list of ec2 buckets'
+ aws ec2 describe-instances
print list of ec2 buckets
{
  "Reservations": [
    {
      "ReservationId": "r-092d8befd54a926f7",
      "OwnerId": "860839673074",
      "Groups": [],
      "Instances": [
        {
          "Architecture": "x86_64",
          "BlockDeviceMappings": [
            {
              "DeviceName": "/dev/sda1",
              "Ebs": {
                "AttachTime": "2025-05-27T09:59:01+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-093be5233381bffff"
              }
            }
          ],
          "ClientToken": "77a097a2-909c-4e1f-a856-8df426a3372f",
          "EbsOptimized": false,
          "EnaSupport": true,
          "Hypervisor": "xen",
          "NetworkInterfaces": [
            {
              "Attachment": {
                "AttachTime": "2025-05-27T09:59:00+00:00",
                "AttachmentId": "eni-attach-00fa74c183d6ef613",
                "DeleteOnTermination": true,
                "DeviceIndex": 0,
                "Status": "attached",
                "NetworkCardIndex": 0
              }
            }
          ]
        }
      ]
    }
  ]
}
```

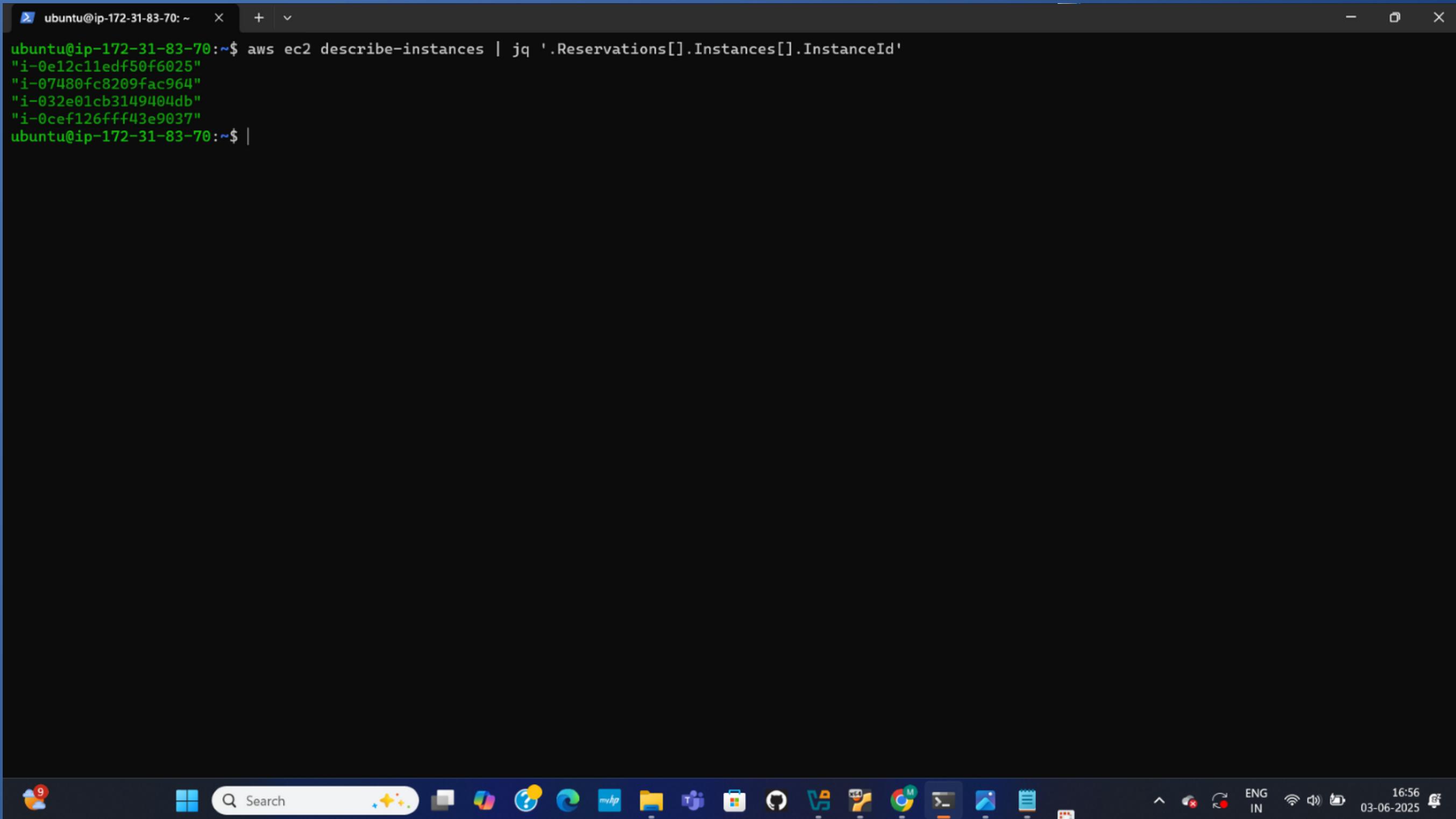
# SCRIPT EXECUTION WITH MORE (CONTINUED OUTPUT)

This page is a continuation of the output from the aws\_resource\_tracker.sh script piped to more. It displays more of the EC2 instance details, including ReservationId, OwnerId, and various instance attributes. It also shows the echo commands for "print list of lambda functions" and "print list of IAM Users" before their respective aws cli commands.

```
ubuntu@ip-172-31-83-70:~ x + v - o x
{
  "ReservationId": "r-092d8befd54a926f7",
  "OwnerId": "860839673074",
  "Groups": [],
  "Instances": [
    {
      "Architecture": "x86_64",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": {
            "AttachTime": "2025-05-27T09:59:01+00:00",
            "DeleteOnTermination": true,
            "Status": "attached",
            "VolumeId": "vol-093be5233381bffffd"
          }
        }
      ],
      "ClientToken": "77a097a2-909c-4e1f-a856-8df426a3372f",
      "EbsOptimized": false,
      "EnaSupport": true,
      "Hypervisor": "xen",
      "NetworkInterfaces": [
        {
          "Attachment": {
            "AttachTime": "2025-05-27T09:59:00+00:00",
            "AttachmentId": "eni-attach-00fa74c183d6ef613",
            "DeleteOnTermination": true,
            "DeviceIndex": 0,
            "Status": "attached",
            "NetworkCardIndex": 0
          },
          "Description": "",
          "Groups": [
            {
              "GroupId": "sg-0ed000d3c8012dd57"
            }
          ]
        }
      ],
      "--More--+
      echo 'print list of lambda functions'
      + aws lambda list-functions
      + echo 'print list of IAM Users'
      + aws iam list-users
  ]
}
```

# FILTERING EC2 INSTANCE IDS WITH JQ

This page demonstrates filtering EC2 instance IDs from the aws ec2 describe-instances command using jq. The command `aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'` successfully extracts and displays four instance IDs: "i-0e12c11edf50f6025", "i-07480fc8209fac964", "i-032e01cb3149404db", and "i-0cef126fff43e9037".

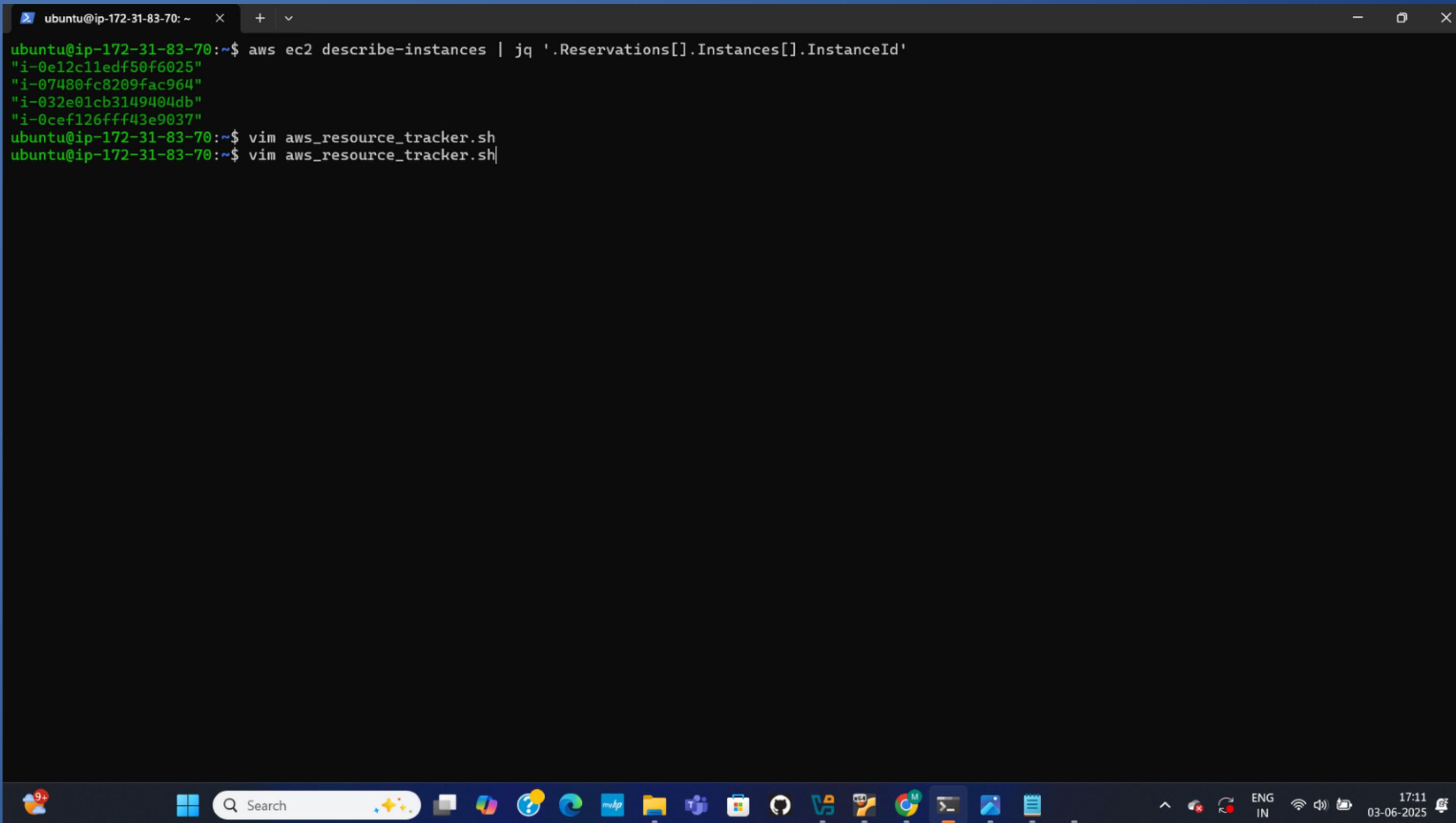


```
ubuntu@ip-172-31-83-70:~$ aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'
"i-0e12c11edf50f6025"
"i-07480fc8209fac964"
"i-032e01cb3149404db"
"i-0cef126fff43e9037"
ubuntu@ip-172-31-83-70:~$ |
```

The screenshot shows a terminal window on a Linux desktop environment. The terminal window has a dark background and contains the command and its output. The desktop taskbar at the bottom shows various application icons, including a file manager, a browser, and system tools. The system tray on the right side of the taskbar displays the date (03-06-2025), time (16:56), battery status, and network connectivity.

## RE-EDITING THE SCRIPT

This page shows the re-accessing and presumably re-editing of the aws\_resource\_tracker.sh script using vim after successfully filtering EC2 instance IDs with jq.



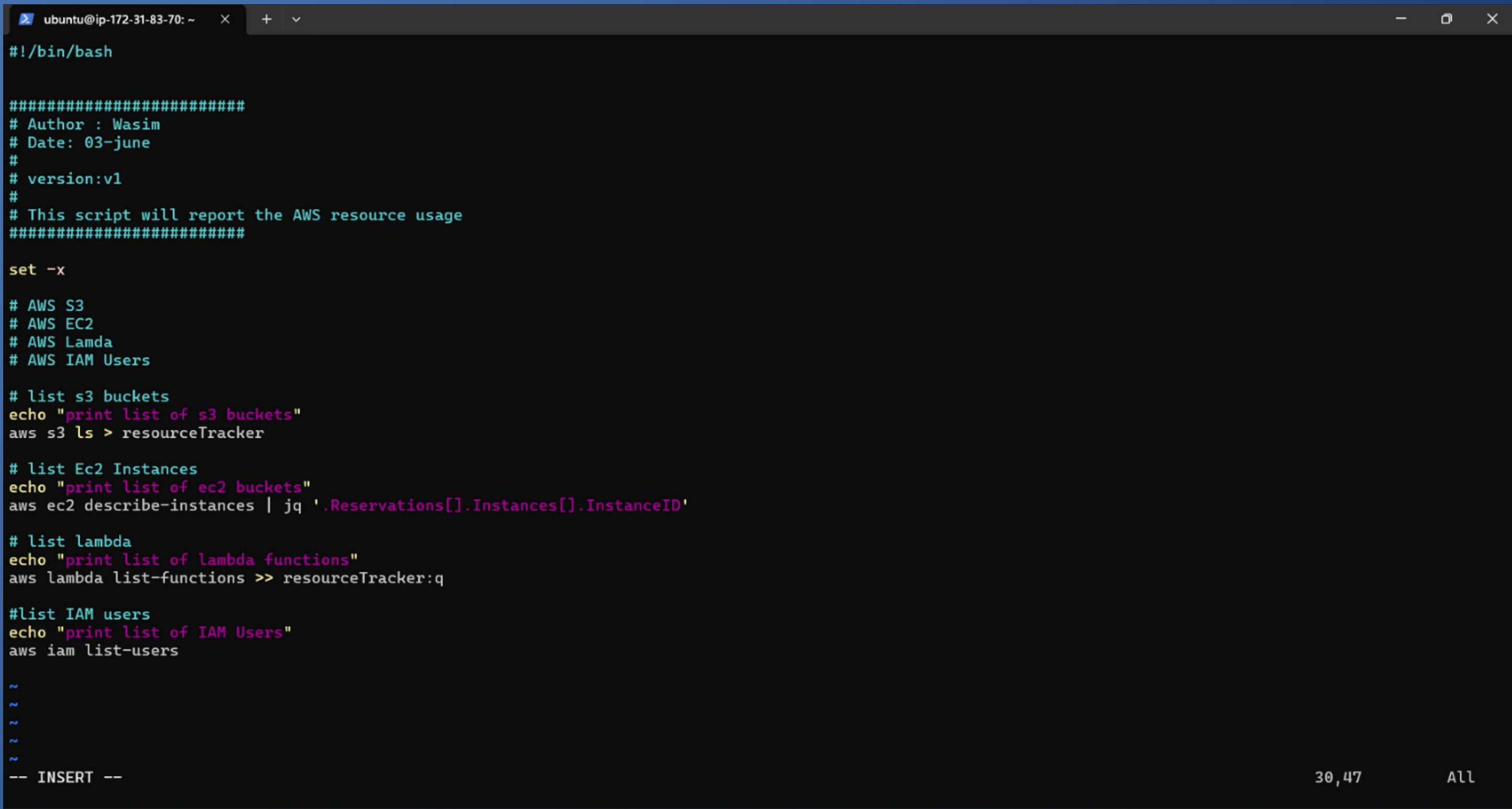
The screenshot shows a terminal window on a Linux desktop environment. The terminal content is as follows:

```
ubuntu@ip-172-31-83-70:~$ aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'
"i-0e12c11edf50f6025"
"i-07480fc8209fac964"
"i-032e01cb3149404db"
"i-0cef126ffff43e9037"
ubuntu@ip-172-31-83-70:~$ vim aws_resource_tracker.sh
ubuntu@ip-172-31-83-70:~$ vim aws_resource_tracker.sh|
```

The desktop taskbar at the bottom shows various application icons, including a file with a red dot, a search bar, and icons for Microsoft Word, Excel, and other productivity tools. The system tray indicates the date as 03-06-2025, the time as 17:11, and language settings as ENG IN.

# MODIFIED SCRIPT - OUTPUT REDIRECTION

This page displays the modified aws\_resource\_tracker.sh script. The script now includes set -x for debugging. The output of aws s3 ls is redirected to a file named resourceTracker (> resourceTracker). The jq filter for EC2 instance IDs is now part of the script (aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'). Additionally, the output of aws lambda list-functions is appended to resourceTracker (>> resourceTracker).



```
ubuntu@ip-172-31-83-70: ~ + - X
#!/bin/bash

#####
# Author : Wasim
# Date: 03-june
#
# version:v1
#
# This script will report the AWS resource usage
#####

set -x

# AWS S3
# AWS EC2
# AWS Lamda
# AWS IAM Users

# list s3 buckets
echo "print list of s3 buckets"
aws s3 ls > resourceTracker

# list Ec2 Instances
echo "print list of ec2 buckets"
aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'

# list lambda
echo "print list of lambda functions"
aws lambda list-functions >> resourceTracker:q

#list IAM users
echo "print list of IAM Users"
aws iam list-users

~
~
~
~
~

-- INSERT --
```