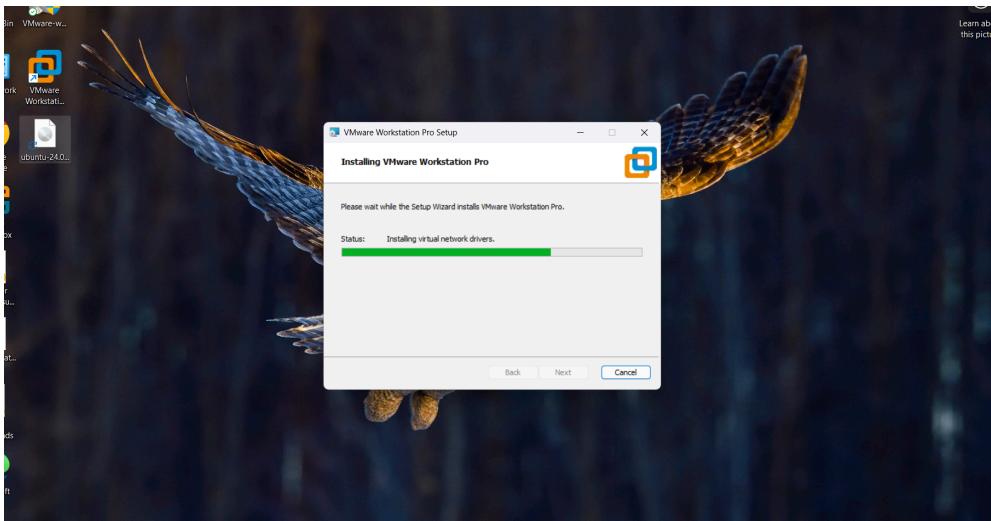
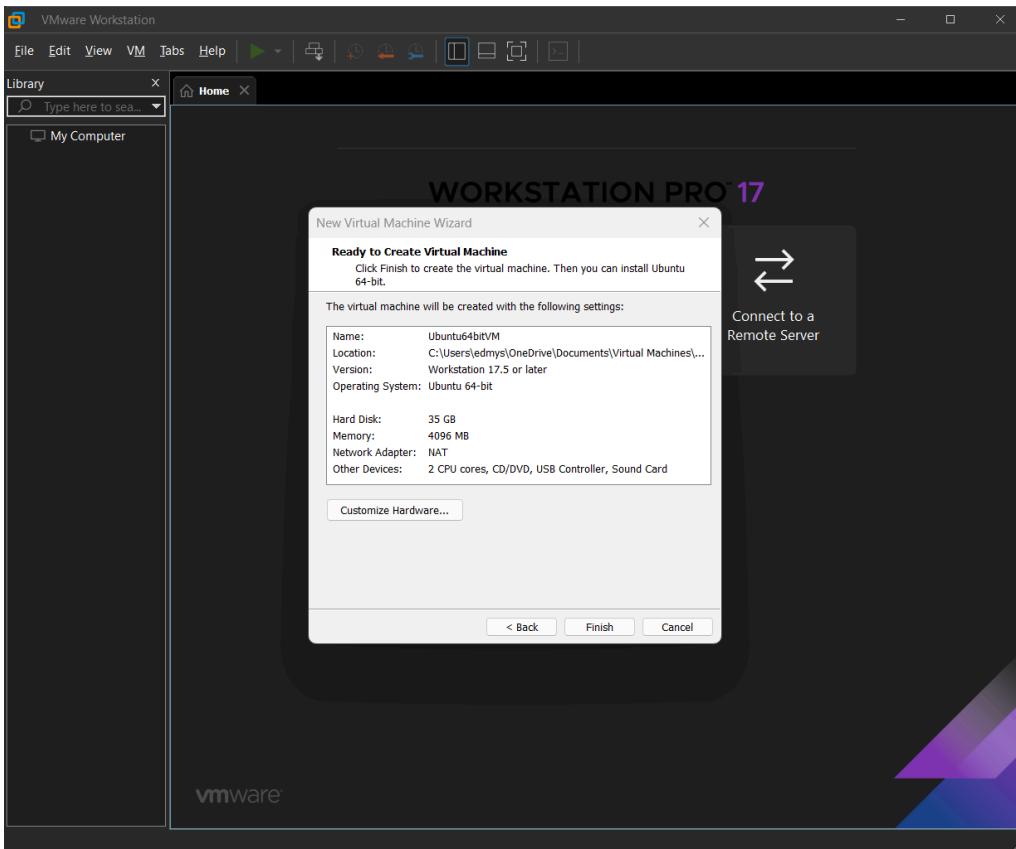


## DELIVERABLES FOR TASK #1

Screenshot of installation in progress on Task 1.1

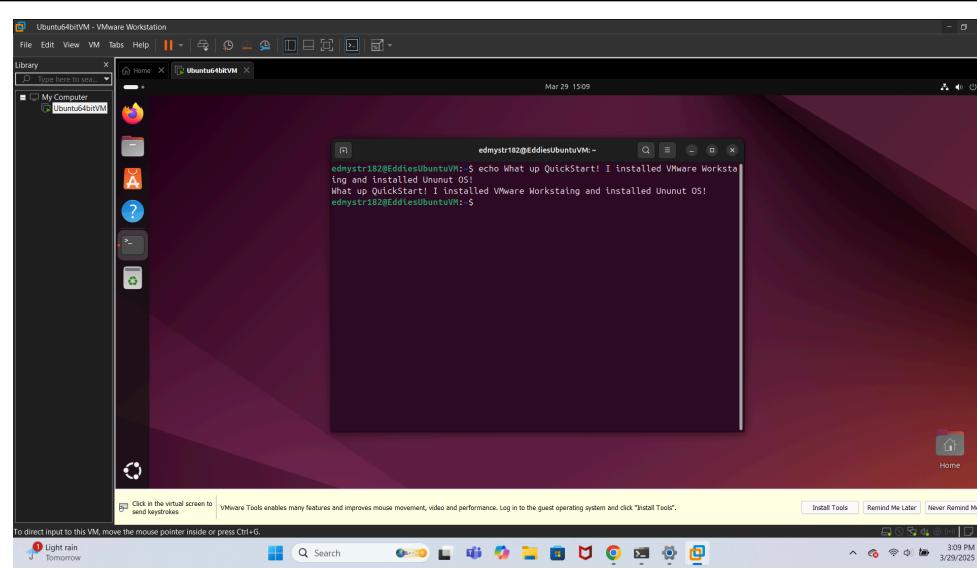
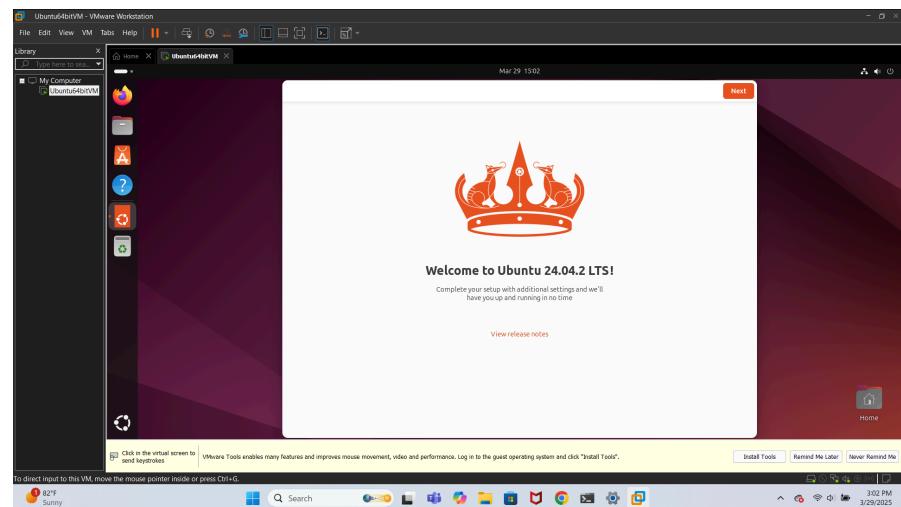
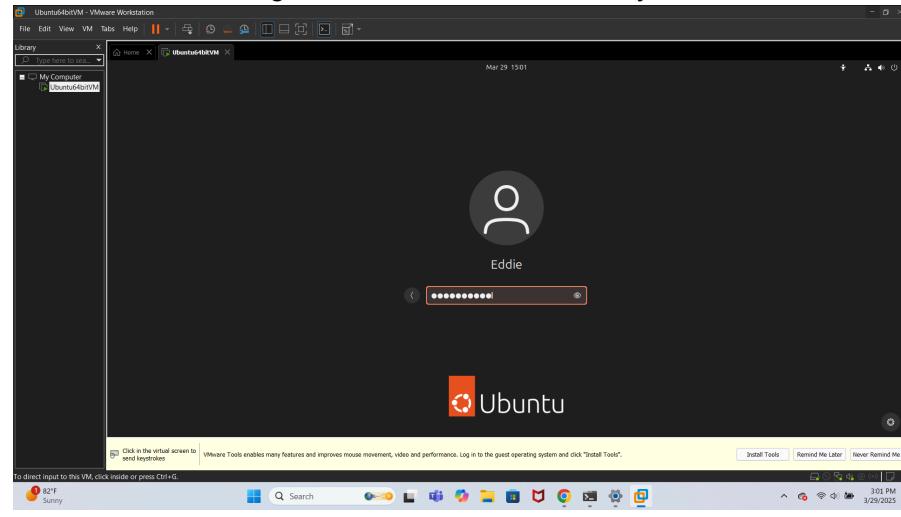


Screenshot of window that shows you the summary of your VM Configuration in Task 1.2



## DELIVERABLES FOR TASK #2

Screenshot of the login screen of Ubuntu where you need to enter the username and password.



## DELIVERABLES FOR TASK #3

### 1. Exercise 1 Tasks:

a.

```
edmystr182@EddiesUbuntuVM:~$ pwd  
/home/edmystr182  
edmystr182@EddiesUbuntuVM:~$
```

edmystr182@ = **Username**

EddiesUbuntuVM = **VM name (Host Name)**

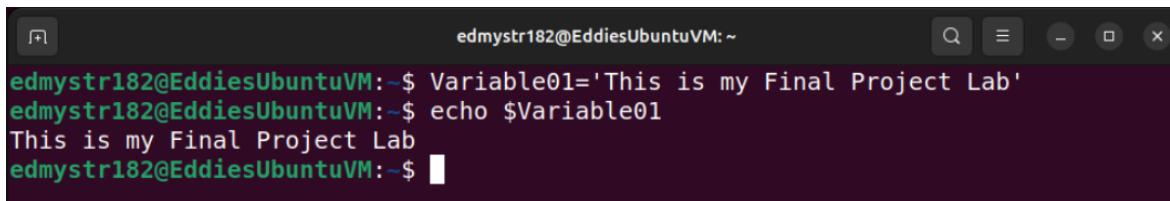
~ = **Home directory (/home/edmystr182)**

b. ls stands for “list” and it lists the files and directories in the current working directory

- ls -a: List hidden files
- ls -l: Displays long listing format: permissions, owner, size, date, and name.
- ls -la: Combines commands -l and -a
- ls -r: Reverses the order of the output (works with other options too)
- ls -R: Lists contents of subdirectories as well, going deep into folder structures.
- ls \*.txt: Only lists files that match the pattern (e.g., all .txt files).

c. A variable is like a label that stores data for use later.

- In the below example the label “Variable01” is given to the data “This is my Final Project Lab”. Then it is called on by the command “echo” and then the data is printed on the screen. It is useful data that is called on often.



```
edmystr182@EddiesUbuntuVM:~$ Variable01='This is my Final Project Lab'  
edmystr182@EddiesUbuntuVM:~$ echo $Variable01  
This is my Final Project Lab  
edmystr182@EddiesUbuntuVM:~$
```

### 2. Exercise 2 Tasks:

a. Define Internal and External commands with examples.

Internal Tasks	External Tasks
<p><b>Definition:</b> Built into the shell (like bash), executed directly by the shell without needing an external program.</p> <p>Faster to execute since they don't require launching a new process.</p> <p><b>Examples:</b></p> <p>cd – Changes the current directory.</p> <p>echo – Displays a message or variable value.</p>	<p><b>Definition:</b> Stored as separate executable files on the system (usually in /bin, /usr/bin, etc.).</p> <p>The shell calls the program from the filesystem.</p> <p><b>Examples:</b></p> <p>ls – Lists directory contents.</p> <p>cat – Displays the contents of a file.</p>

b. Type -a echo

```
edmystr182@EddiesUbuntuVM:~$ type -a echo
echo is a shell builtin
echo is /usr/bin/echo
echo is /bin/echo
edmystr182@EddiesUbuntuVM:~$
```

These results mean that the echo command is built into the shell, but there is also an external binary version at /usr/bin/echo and /bin/echo

- c. Double quoting: Allows variable expansion, command substitution, and escape sequences like \n. It is useful when you want to include variable values in strings.

```
edmystr182@EddiesUbuntuVM:~$ echo "This is the Linux Essential Class"
This is the Linux Essential Class
edmystr182@EddiesUbuntuVM:~$
```

- d. Single Quoting: Everything is taken literally—no variable expansion, no command substitution. It is useful when you want to preserve exact content.
- e. By using single quotes. If you use double quotes the command will read the '\$' as a variable.

```
edmystr182@EddiesUbuntuVM:~$ echo 'The Car Cost $1000'
The Car Cost $1000
edmystr182@EddiesUbuntuVM:~$ echo "The Car Cost $1000"
The Car Cost 000
edmystr182@EddiesUbuntuVM:~$
```

- f. The **backslash (\)** is used in Bash as an escape character, meaning it tells the shell to treat the next character literally.

**Single Quotes:** The backslash loses its meaning and will be printed with the data.

**Double Quotes:** Backslash can escape certain characters: \$, ", ` , \, and newline. Useful to include those special characters in a string.

- g.

```
edmystr182@EddiesUbuntuVM:~$ echo Today is `date`
Today is Sun Mar 30 01:40:20 PM CDT 2025
edmystr182@EddiesUbuntuVM:~$
```

### 3. Exercise 3 Tasks:

- a. **Definition:** Control statements are special syntax in Bash that control the flow of command execution. They determine when and how commands run, especially in sequences, conditions, and loops.
- b. **Basic Control Statements:**
  - (;) Semicolon: Run multiple commands one after another, regardless of success.
  - (&&) Double Ampersand: Run the next command only if the previous one succeeds.
  - (||) Double Pipe: Runs the next command only if the previous one fails
  - (if) If Statement: Executes commands based on a true/false condition.
  - (elif) Else If Statement: Adds more conditions to an if block.
  - (else) Else Statement: Executes if none of the previous if or elif conditions were true.
  - (for) For Loop: Repeats commands for a list of items.
  - (while) While Loop: Repeats commands while a condition is true.
- c. **Example of (;**): The below command created a directory, moved into the directory, and created a .txt file in that new directory by using (;) between commands.

```
Mar 30 14:19  
edmystr182@EddiesUbuntuVM:~/Projects  
edmystr182@EddiesUbuntuVM:~/Projects$ mkdir Projects; cd Projects; touch Hello.txt  
Hello.txt  
edmystr182@EddiesUbuntuVM:~/Projects$
```

Example of `(&&)`: The below command printed the current directory, then if `(pwd)` succeeds used `(echo)` to say “This is your current working directory”

```
edmystr182@EddiesUbuntuVM:~$ pwd && echo "This is your current working directory"  
/home/edmystr182  
This is your current working directory  
edmystr182@EddiesUbuntuVM:~$
```

Example of `(||)`: The below command attempts to change directories into a directory that does not exist, the `||` then prints (if the command fails) “Directory does not exist”

```
edmystr182@EddiesUbuntuVM:~$ ls  
Desktop Documents Downloads Music Pictures Projects Public snap Templates Videos  
edmystr182@EddiesUbuntuVM:~$ cd FakeDirecotry || echo "Directory does not exist"  
bash: cd: FakeDirecotry: No such file or directory  
Directory does not exist  
edmystr182@EddiesUbuntuVM:~$
```

Bonus `(&&)` with `(||)` together: The below command uses both and prints “Succeeded” if the command works and “Failed” if it does not.

```
edmystr182@EddiesUbuntuVM:~$ cd Music && echo "Succeeded" || echo "Failed"  
Succeeded  
edmystr182@EddiesUbuntuVM:~/Music$ cd NotRealDirectory && echo "Succeeded" || echo "Failed"  
bash: cd: NotRealDirectory: No such file or directory  
Failed  
edmystr182@EddiesUbuntuVM:~/Music$
```

## DELIVERABLES FOR TASK #4

1. What is the Administrator account?
  - a. The administrator account refers to the root user. The root user is the superuser with full control over the system. The root user can install software, modify system files, create/delete users, and override permissions.
2. What is meant by switching accounts?
  - a. You are usually not logged in directly as root. Instead, use the `(sudo)` command. This runs a command with root privileges, if your user is in the sudo group.
3. What are system accounts?
  - a. System accounts are user accounts created by the system, not by human users. They are used to run background services, daemons, and handle specific system tasks.
  - b. Key Characteristics:
    - Usually have user IDs (UIDs) below 1000 (on most Linux distros).
    - Do not have a home directory or login shell (or use `/usr/sbin/nologin` or `/bin/false`).
    - Not meant for interactive login.
    - Help isolate processes and improve security by assigning services their own identities.
  - c. Common System Accounts:
    - **root**: Superuser / Admin
    - **nobody**: Used for processes with the least privilege
    - **www-data**: Runs the web server (e.g., Apache, Nginx)
    - **daemon**: Generic account for daemons
    - **sshd**: Runs the SSH service

➤ **systemd-network**: Manages network services

4. What does the (su) command do?

- The su command stands for "substitute user". It is usually used to switch to root.

```
root@EddiesUbuntuVM: /home/edmystr182
edmystr182@EddiesUbuntuVM:~$ su
Password:
root@EddiesUbuntuVM:/home/edmystr182# pwd
/home/edmystr182
root@EddiesUbuntuVM:/home/edmystr182#
```

5. The who command in Linux is used to show who is currently logged into the system.

**What It Displays:**

- Username of logged-in users
- Terminal (TTY) they're using
- Date and time they logged in
- Possibly, their IP address or hostname

```
edmystr182@EddiesUbuntuVM:~$ who
edmystr182 seat0      2025-03-30 17:19 (login screen)
edmystr182 tty2      2025-03-30 17:19 (tty2)
edmystr182@EddiesUbuntuVM:~$ whoami
edmystr182
```

6. (**w**) command: Shows a detailed overview of who is logged in and what they are doing on the system.

(**last**) command: Shows a history of logins on the system — including user logins, reboots, and shutdowns.

```
edmystr182@EddiesUbuntuVM:~$ w
17:31:19 up 12 min,  1 user,  load average: 0.00, 0.18, 0.30
USER   TTY      FROM             LOGIN@   IDLE    JCPU   PCPU WHAT
edmystr1 tty2      -          17:19   12:28   0.13s  0.11s /usr/libexec/gn
edmystr182@EddiesUbuntuVM:~$ last
edmystr1 tty2      tty2      Sun Mar 30 17:19  still logged in
edmystr1 seat0      login screen  Sun Mar 30 17:19  still logged in
reboot  system boot 6.11.0-21-generi Sun Mar 30 17:18  still running
reboot  system boot 6.11.0-21-generi Sun Mar 30 16:58  still running
reboot  system boot 6.11.0-21-generi Sun Mar 30 16:54 - 16:58 (00:03)
edmystr1 tty2      tty2      Sun Mar 30 13:18 - down (03:36)
edmystr1 seat0      login screen  Sun Mar 30 13:18 - down (03:36)
reboot  system boot 6.11.0-21-generi Sun Mar 30 13:16 - 16:54 (03:37)
reboot  system boot 6.11.0-21-generi Sun Mar 30 13:14 - 13:16 (00:01)
edmystr1 tty2      tty2      Sun Mar 30 12:31 - down (00:42)
edmystr1 seat0      login screen  Sun Mar 30 12:31 - down (00:42)
reboot  system boot 6.11.0-21-generi Sun Mar 30 12:31 - 13:14 (00:43)
edmystr1 tty2      tty2      Sat Mar 29 18:20 - crash (18:10)
edmystr1 seat0      login screen  Sat Mar 29 18:20 - crash (18:10)
reboot  system boot 6.11.0-21-generi Sat Mar 29 18:20 - 13:14 (18:54)
edmystr1 tty2      tty2      Sat Mar 29 15:01 - down (03:18)
edmystr1 seat0      login screen  Sat Mar 29 15:01 - down (03:18)
reboot  system boot 6.11.0-21-generi Sat Mar 29 14:55 - 18:19 (03:24)

wtmp begins Sat Mar 29 14:55:21 2025
edmystr182@EddiesUbuntuVM:~$
```

## Bonus Work: Personalization and Scripting

### Objective #1: Create a personalized welcome message

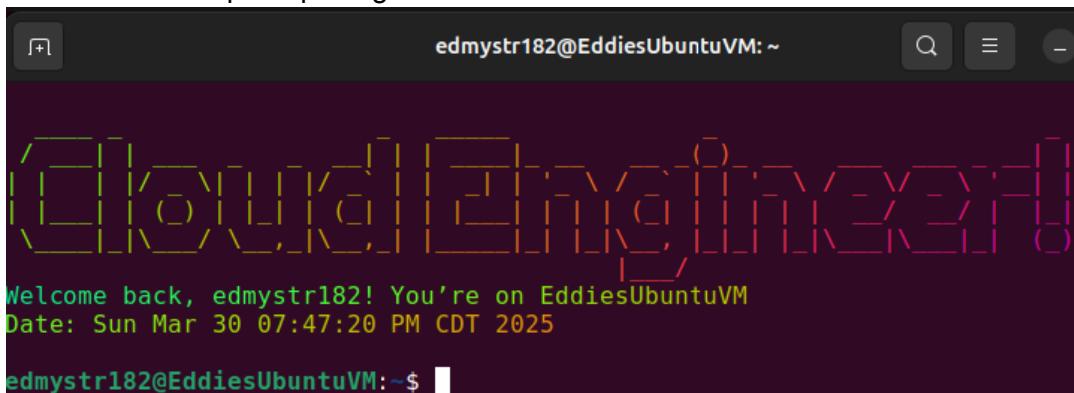
Steps:

1. Download figlet, toilet, and lolcat
  - a. sudo apt install figlet (Bubble Letters)
  - b. sudo apt install toilet (Colors and Effects)
  - c. sudo apt install lolcat (Rainbow Colors)
2. Create Welcome Script
  - a. Run command: "nano ~/.bashrc" and add to the bottom

```
# === Custom Welcome Banner ===
echo
figlet "Cloud Engineer!" | lolcat
echo "Welcome back, $USER! You're on $(hostname)" | lolcat
echo "Date: $(date)" | lolcat
echo
```

b.

3. Show welcome banner upon opening terminal



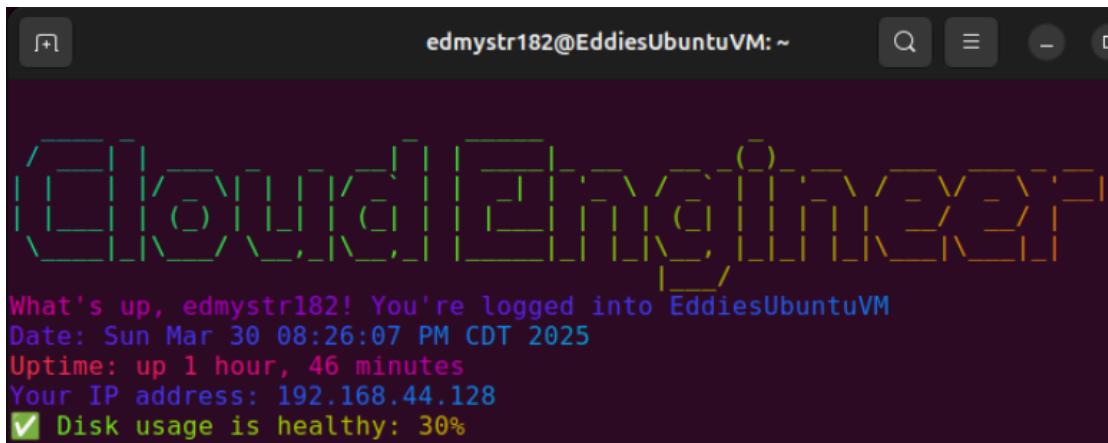
The terminal window shows a custom welcome banner. At the top, it says "edmystr182@EddiesUbuntuVM: ~". Below that is a decorative banner made of various ASCII characters like brackets and underscores. Underneath the banner, the text "Welcome back, edmystr182! You're on EddiesUbuntuVM" and "Date: Sun Mar 30 07:47:20 PM CDT 2025" is displayed in green. At the bottom, there is a prompt "edmystr182@EddiesUbuntuVM:~\$".

### Objective #2: Check disc space and add a warning if disc space is over a 80% Threshold.

1. Add the below script to ~/.bashrc

```
# === Disk Space Check ===
THRESHOLD=80
USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ "$USAGE" -gt "$THRESHOLD" ]; then
    echo "⚠️ Warning: Disk usage is above ${THRESHOLD}% (${USAGE}%)" | lolcat
else
    echo "✅ Disk usage is healthy: ${USAGE}%" | lolcat
fi
```



The terminal window shows disk space usage. At the top, it says "edmystr182@EddiesUbuntuVM: ~". Below that is a decorative banner. Underneath the banner, the text "What's up, edmystr182! You're logged into EddiesUbuntuVM", "Date: Sun Mar 30 08:26:07 PM CDT 2025", "Uptime: up 1 hour, 46 minutes", "Your IP address: 192.168.44.128", and "✅ Disk usage is healthy: 30%" is displayed in blue. The "Disk usage is healthy: 30%" line has a checkmark icon before it.

