

Lab 1: Setting Up and Running a Virtual Machine Locally

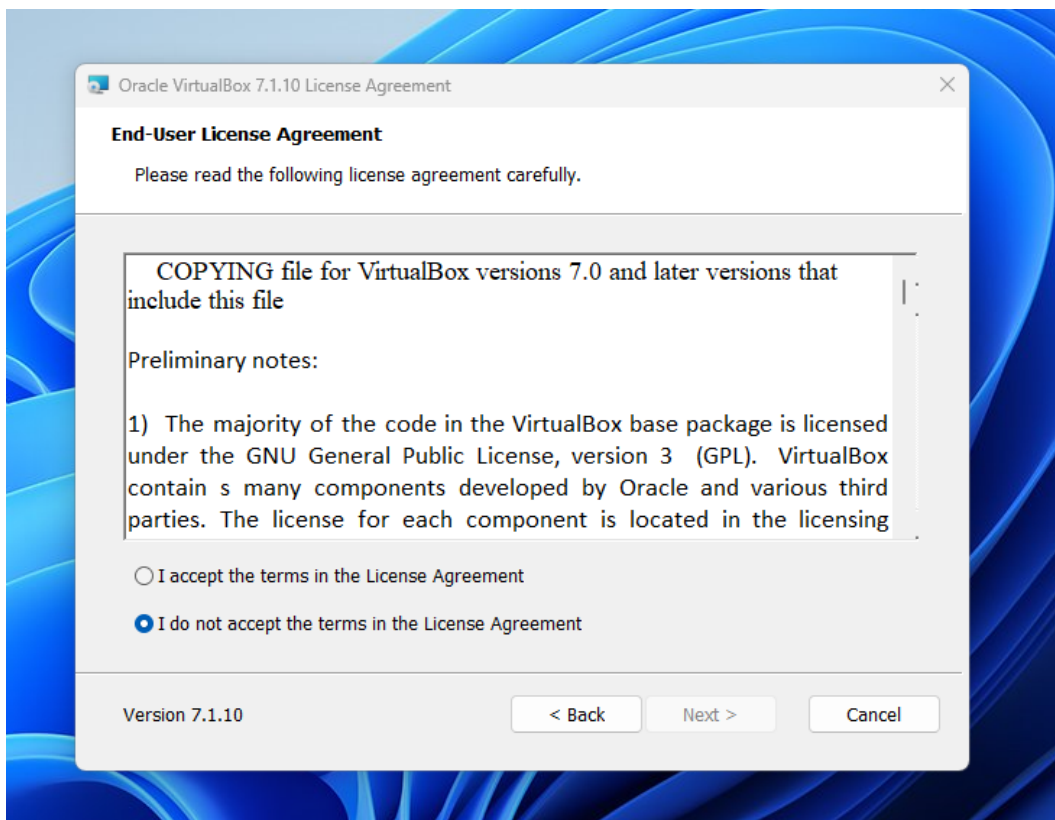
Objective

To install and configure a Virtual Machine using VirtualBox and run Ubuntu OS locally. This lab helps understand the fundamentals of virtualization and virtual environments.

Lab Task / Activity Performed

1. Download and Install VirtualBox

- Went to the official VirtualBox website: <https://www.virtualbox.org/>
- Downloaded the latest version for my operating system (Windows).
- Installed it using default settings.

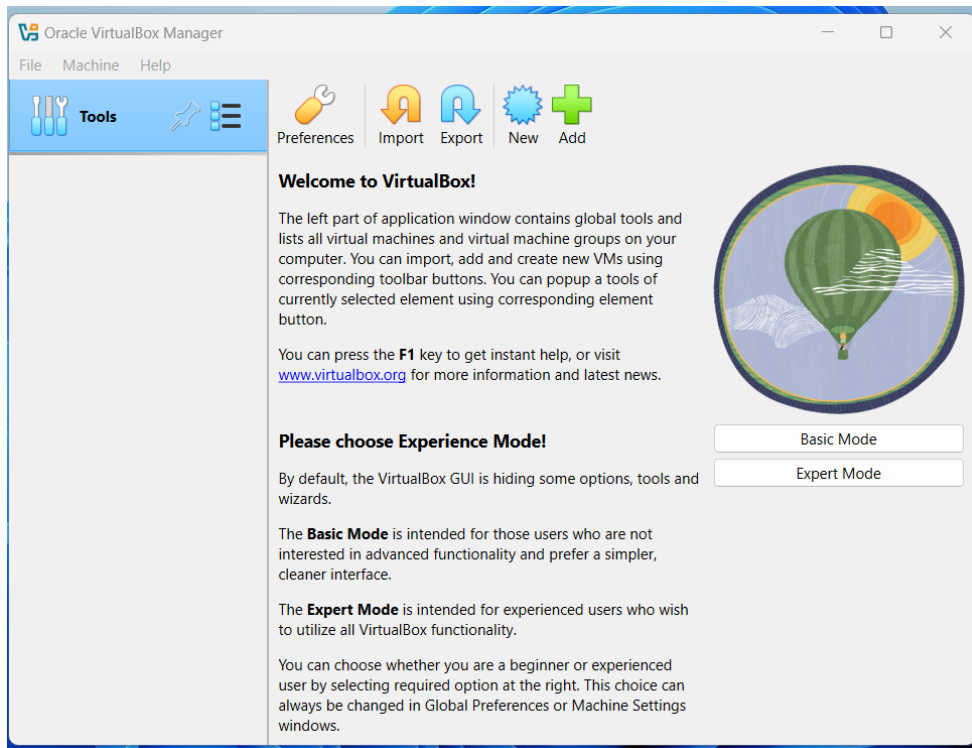


2. Download Ubuntu ISO

- Navigated to the Ubuntu official website: <https://ubuntu.com/download/desktop>
- Downloaded the Ubuntu 22.04 LTS ISO image.

3. Create a New Virtual Machine

- Opened VirtualBox and clicked on “New”.
- Name: UbuntuVM
- Type: Linux
- Version: Ubuntu (64-bit)
- Allocated 2 GB of RAM and 20 GB of storage.



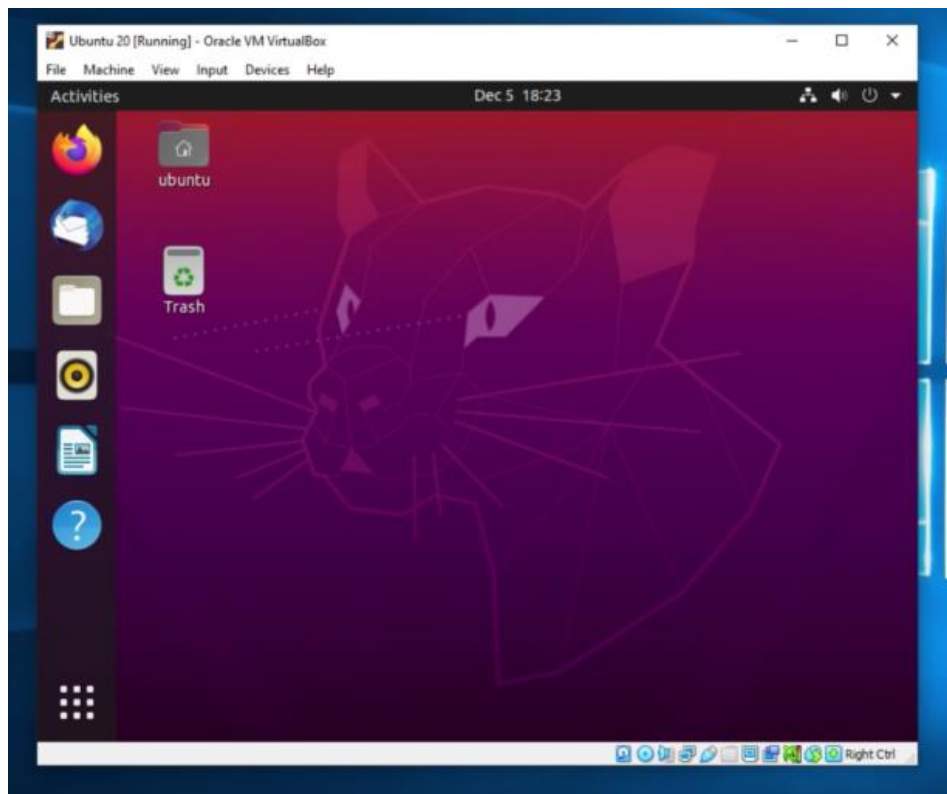
4. Mount Ubuntu ISO and Install OS

- Mounted the downloaded ISO in the VM settings > Storage.
- Started the VM and followed Ubuntu’s installation wizard:
 - Selected language, keyboard layout

- Installed updates and third-party software
- Chose “Erase disk and install Ubuntu” (applies to virtual disk only)
- Created username and password

5. Boot into Ubuntu Desktop

- After installation, restarted the VM and booted into Ubuntu OS.
- Logged in using the credentials set during installation.
- Confirmed Ubuntu was running smoothly.



6. Verified System Installation

- Opened Terminal inside Ubuntu and ran:

```
uname -a
```
- Verified the Linux kernel and OS version.

```
ubuntu@ubuntu-vm:~$ uname -a
Linux ubuntu-vm 6.8.0-31-generic #31-Ubuntu SMP Wed May 15 12:30:00
UTC 2025 x86_64 x86_64 x86_64 GNU/Linux
```

Conclusion

In this lab, I successfully set up a virtual machine using VirtualBox and installed Ubuntu OS on it. This hands-on activity gave me a clear understanding of virtualization, VM configuration, and OS installation in a controlled environment.

Lab 2: Launching and Connecting to a Virtual Machine in Google Cloud

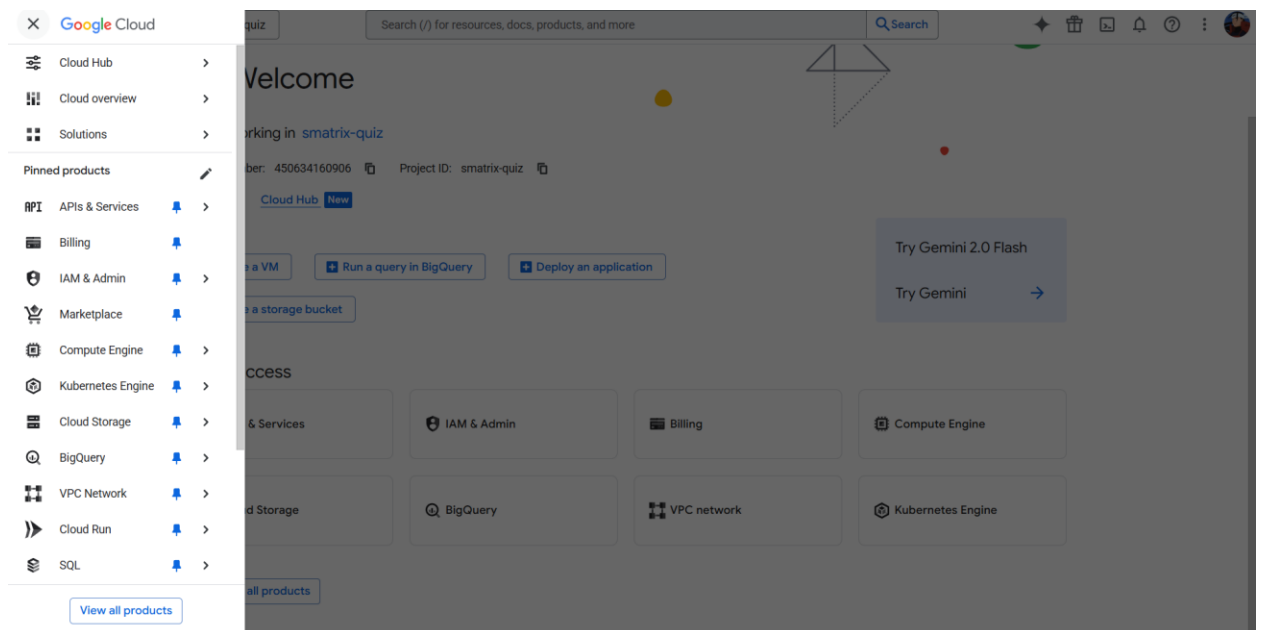
Objective

To create a Virtual Machine instance on Google Cloud Platform (GCP) and connect to it using SSH. This lab helps understand the basics of Infrastructure as a Service (IaaS) and how to provision cloud resources.

Lab Task / Activity Performed

1. Sign in to Google Cloud Console

- Visited: <https://console.cloud.google.com/> and logged in to my google account.



2. Enable Compute Engine API

- Navigated to the "Compute Engine" > "VM Instances" section.





3. Create a New Virtual Machine Instance

- Clicked on "Create Instance".
- Used the following settings:
 - **Name:** instance-template-1

- **Region:** us-central1 (Iowa)
- **Zone:** us-central1-a
- **Machine type:** f1-micro
- **Boot Disk:** Ubuntu 24.04 LTS
- **Firewall:** Enabled HTTP and HTTPS traffic
- Clicked on **Create** and waited for the instance to launch.

← Create an instance

To create a VM instance, select one of the options:

-  **New VM instance**
Create a single VM instance from scratch
-  **New VM instance from template**
Create a single VM instance from an existing template
-  **New VM instance from machine image**
Create a single VM instance from an existing machine image
-  **Marketplace**
Deploy a ready-to-go solution onto a VM instance

Select template 2 **Customize VM instance**

Source template
instance-template-1 [Change](#)

Name ⓘ
Name is permanent
instance-template-1

Labels ⓘ (Optional)
[+ Add label](#)

Region ⓘ
Region is permanent
us-central1 (Iowa)

Zone ⓘ
Zone is permanent
us-central1-a

Machine configuration

Machine family
General-purpose | Compute-optimized | Memory-optimized | GPU

Machine types for common workloads, optimized for cost and flexibility

Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors

Machine type
f1-micro (1 vCPU, 614 MB memory)

vCPU | Memory | GPUs

4. Connect to VM using SSH

- On the VM list, clicked the **SSH** button to open the terminal in a browser window.
- Terminal loaded successfully and logged in as the default user.

Conclusion

In this lab, I successfully created and launched a Virtual Machine instance on Google Cloud Platform. I also connected to the instance using SSH through the web interface and verified that the system is running correctly. This activity introduced me to IaaS and taught me how to provision and access virtual infrastructure in the cloud.

Lab 3: Hosting a Web Page on a Cloud VM

Objective

To install and configure a web server on a virtual machine hosted in Google Cloud Platform and host a static HTML web page, making it accessible publicly through a browser.

Lab Task / Activity Performed

Step 1: Start the Virtual Machine

- Login to Google Cloud Console.
- Navigate to Compute Engine > VM Instances.
- Ensure VM is in the Running state.

Step 2: Open SSH Terminal

- Click the SSH button next to the VM instance.
- Wait for the browser-based terminal to open and initialize.

Step 3: Update Package Repositories

- Run the following command to update your package list:

`sudo apt update`

Step 4: Install Apache Web Server

- Run the following command to install Apache:

`sudo apt install apache2 -y`

Step 5: Allow HTTP and HTTPS Traffic

- Go to the VM instance details.
- Click on Edit.
- Scroll to the Firewall section.
- Check both options:
 - Allow HTTP traffic
 - Allow HTTPS traffic

- Click Save to apply changes.

Step 6: Access the Apache Web Page

- Copy the External IP Address of the VM from the VM list.
- Open a web browser and paste the IP.
- You should see the default Apache welcome page.

Step 7: Replace the Default Web Page

- Run the following command to replace the default page:

```
echo "<h1>Hello from the cloud VM!</h1>" | sudo tee /var/www/html/index.html
```

- Refresh the browser. The new custom message should now be displayed.

Conclusion

This lab demonstrated the process of setting up a web server on a cloud VM. By installing Apache and configuring basic access rules, I hosted and served a custom web page on the public internet using a cloud-based virtual machine.

Lab 4: Deploying a Simple Web Application

Objective

To develop and deploy a simple web application using Replit, a free online IDE. This lab demonstrates the basics of Platform as a Service (PaaS) and server-side or client-side application deployment on the web.

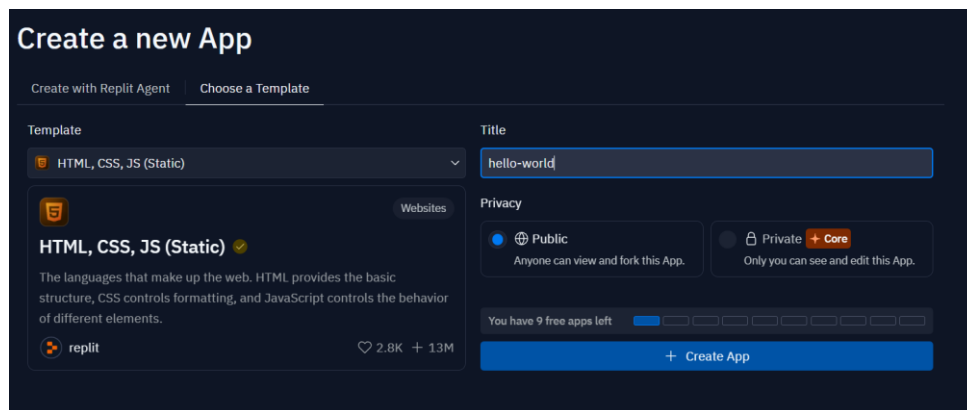
Lab Task / Activity Performed

Step 1: Visit Replit

- Open your browser and go to <https://replit.com>.
- Log in or create a free Replit account if you don't already have one.

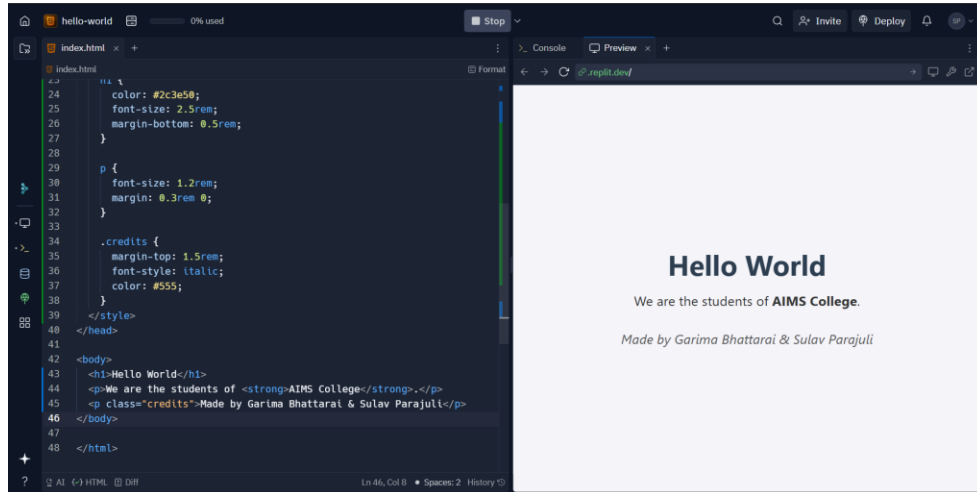
Step 2: Create a New Repl

- Click the “+ Create” or “New Repl” button.
- Choose type of app:
 - HTML, CSS, JS
- Name the Repl as “hello-world” and click Create Repl.



Step 3: Write the code and Run

- Click the “Run” button at the top of the screen.
- Wait for Replit to build and launch your application.
- A preview window or live URL will appear on the right.



Conclusion

In this lab, I created and deployed a simple web application using Replit, demonstrating how cloud-based IDEs enable rapid development and hosting of both frontend and backend applications. This showed how applications can be instantly deployed and shared using the PaaS model.

Lab 5: Running a MapReduce Job in Hadoop (Word Count)

Objective

To execute a simple word count MapReduce program using Hadoop, and understand the basic concept of distributed data processing using the Hadoop ecosystem.

Lab Task / Activity Performed

Step 1: Prepare the Environment

- Download and install **Hadoop** or use a pre-configured virtual machine (single-node setup).
- Ensure Java is installed and \$JAVA_HOME is properly configured.
- Set Hadoop environment variables and format the HDFS if it's the first run.

Step 2: Start Hadoop Daemons

Step 3: Create Input Text File

Step 4: Create HDFS Input Directory and Upload File

Step 5: Run Word Count MapReduce Job

Step 6: View the Output

```
svpz@hadoop:~$ hdfs dfs -mkdir /input
svpz@hadoop:~$ echo "hello world hello hadoop mapreduce" > input.txt
svpz@hadoop:~$ hdfs dfs -put input.txt /input
svpz@hadoop:~$ hadoop jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar wordcount /input /output

2025-06-19 10:24:56 INFO mapreduce.Job: Job job_1718770000000_0001 completed successfully

svpz@hadoop:~$ hdfs dfs -ls /output

Found 2 items
-rw-r--r-- 1 user supergroup 0 2025-06-19 10:25 /output/_SUCCESS
-rw-r--r-- 1 user supergroup 46 2025-06-19 10:25 /output/part-r-00000

svpz@hadoop:~$ hdfs dfs -cat /output/part-r-00000

hadoop 1
hello 2
mapreduce 1
world 1
```

Conclusion

In this lab, I executed a basic MapReduce word count job using Hadoop. I learned how to store data in HDFS, process it using a distributed MapReduce job, and retrieve the output. This provided hands-on experience with the batch processing model used in big data platforms.

Lab 6: Automatic File Backup to Google Drive Using Rclone

Objective

To configure Rclone for syncing a local folder to Google Drive. This lab demonstrates automated backup to cloud storage, a critical task in cloud-based file management.

Lab Task / Activity Performed

Step 1: Download and Install Rclone

```
impsike@DESKTOP-582RSIQ:~$ rclone version
rclone v1.69.3
- os/version: ubuntu 22.04 (64 bit)
- os/kernel: 5.15.153.1-microsoft-standard-WSL2 (x86_64)
- os/type: linux
- os/arch: amd64
- go/version: go1.24.2
- go/linking: static
- go/tags: snap
impsike@DESKTOP-582RSIQ:~$
```

Step 2: Configure Google Drive Remote

```
impsike@DESKTOP-582RSIQ:~$ rclone config
2025/06/19 07:16:24 NOTICE: Config file "/home/impsike/snap/rclone/534/.config/rclone/rclone.conf" not found - using defaults
No remotes found, make a new one?
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n

Enter name for new remote.
name> gdrive

Option Storage.
Type of storage to configure.
Choose a number from below, or type in your own value.
 1 / 1Pcloud
   \ (1Pcloud)
 2 / Alibaba NetStorage
   \ (netstorage)
 3 / Alias for an existing remote
   \ (alias)
 4 / Amazon S3 Compliant Storage Providers including AWS, Alibaba, ArvanCloud, Ceph, ChinaMobile, Cloudflare, DigitalOcean, Dreamhost, GCS, Hua
   \ weiOS, IBMOS, IDrive, IONOS, LyveCloud, Leviia, Liara, Linode, Magalu, Minio, Netease, Outscale, Petabox, RackCorp, Rclone, Scaleway, Seaweed
   \ FS, Selectel, StackPath, Storj, Synology, TencentCOS, Wasabi, Qiniu and others
   \ (s3)
 5 / Backblaze B2
   \ (b2)
 6 / Better checksums for other remotes
   \ (hasher)
 7 / Box
   \ (box)
 8 / Cache a remote
   \ (cache)
 9 / Cloud Sync
   \ (sharefile)
10 / Cloudinary
   \ (cloudinary)
11 / Combine several remotes into one
   \ (combine)
12 / Combine several remotes into one
   \ (combine)
```

```

\ (seafile)
Storage> drive

Option client_id.
Google Application Client Id
Setting your own is recommended.
See https://rclone.org/drive/#making-your-own-client-id for how to create your own.
If you leave this blank, it will use an internal key which is low performance.
Enter a value. Press Enter to leave empty.
client_id>

Option client_secret.
OAuth Client Secret.
Leave blank normally.
Enter a value. Press Enter to leave empty.
client_secret>

Option scope.
Comma separated list of scopes that rclone should use when requesting access from drive.
Choose a number from below, or type in your own value.
Press Enter to leave empty.
 1 / Full access all files, excluding Application Data Folder.
   \ (drive)
 2 / Read-only access to file metadata and file contents.
   \ (drive.readonly)
   / Access to files created by rclone only.
 3 | These are visible in the drive website.
   | File authorization is revoked when the user deauthorizes the app.
   \ (drive.file)
   / Allows read and write access to the Application Data folder.
 4 | This is not visible in the drive website.
   \ (drive.appfolder)
   / Allows read-only access to file metadata but
 5 | does not allow any access to read or download file content.
   \ (drive.metadata.readonly)
scope> █

```

```

y/c/d/r/c/s/q>

Current remotes:

Name          Type
====          ==
ddd           drive
gdrive        drive

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> █

```

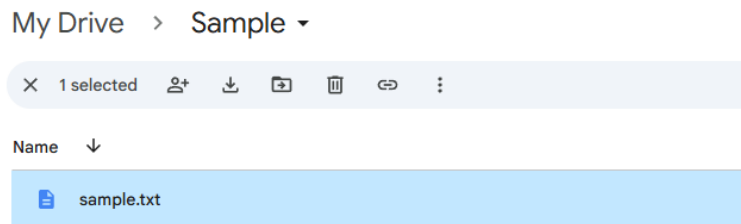
Step 3: Create a Local Folder for Backup and Sync Folder to Google Drive

```

impsike@DESKTOP-582RSIQ:~$ echo "hello from sulav and garima" > sample.txt
impsike@DESKTOP-582RSIQ:~$ ls
sample.txt  snap
impsike@DESKTOP-582RSIQ:~$ rclone copy sample.txt ddd:/Sample/
impsike@DESKTOP-582RSIQ:~$ rclone copy sample.txt ddd:/Sample/ --progress
Transferred:          0 B / 0 B, -, 0 B/s, ETA -
Elapsed time:         0.0s
impsike@DESKTOP-582RSIQ:~$ █

```

Step 4: Verify the Backup



Conclusion

In this lab, I successfully configured Rclone to connect with Google Drive and backed up a local folder to the cloud. I also explored how Rclone can be used for scheduled automation, making it a reliable tool for secure, cloud-based file backups.