**SCP vs RSYNC**

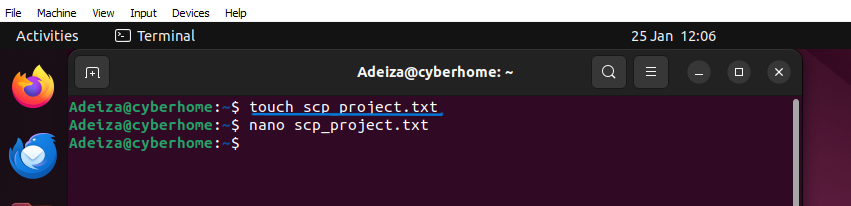
Secure Copy Protocol (SCP) and Remote Synchronization (RSYNC) are two popular file transfer tools used in Linux and Unix-based systems. Both tools provide secure file transfer over a network using the SSH protocol, but they have some differences in terms of functionality and performance.

SCP is a simple and straightforward file transfer tool that provides basic functionality for transferring files between local and remote machines. It is a command-line utility that is included in most SSH clients and provides basic options for specifying the source and destination paths, as well as the user and host to connect to (IP address).

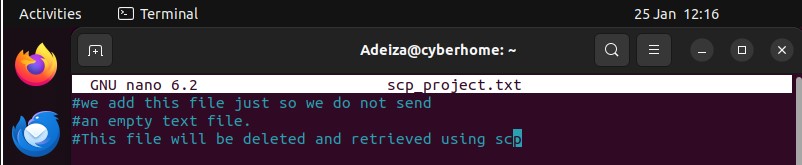
RSYNC, on the other hand, is a more advanced file transfer tool that provides additional functionality for synchronizing directories and preserving file attributes. It is also a command-line utility that is included in most Linux distributions and provides advanced options for specifying the transfer mode, compression, and filtering rules.

**TRANSFERING FILE USING SCP**

* Create a file using the text editor or the “touch” command



* Nano the text file just so it is not empty.



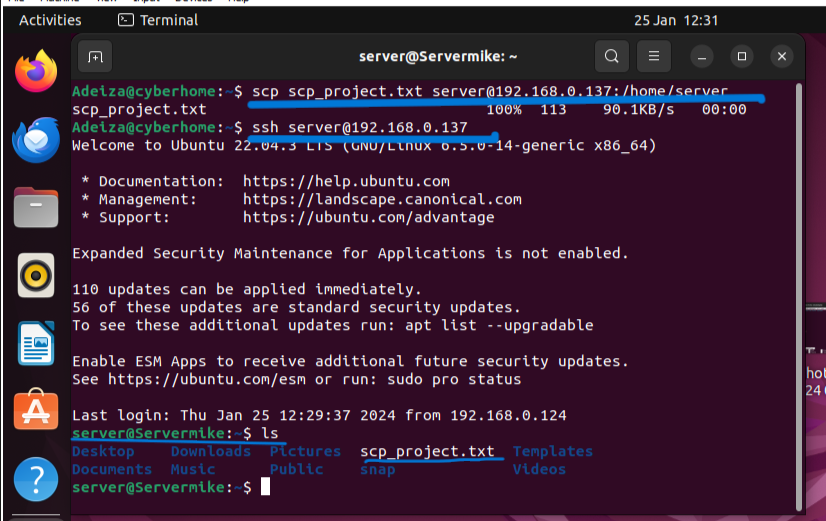
1. **Identify the File**: Identify the file you want to transfer, such as **scp\_project.txt**.
2. **Specify the User and IP Address**: Use the **scp** command to transfer the file to the remote machine, specifying the user and IP address of the remote machine. scp scp\_project.txt user@remote\_ipaddr:/directory

Where:

* + **scp\_project.txt** is the file you want to transfer
  + **user** is the user account on the remote machine
  + **remote\_ipaddr** is the IP address of the remote machine
  + **/directory** is the location where you want to save the file on the remote machine

1. **Specify the Location**: Use the colon (**:**) to specify the location where you want to save the file on the remote machine. For example, if it is a configuration file, you may want to save it in the **/etc/** directory.

After transferring the file, use **ssh** to log in to the remote server and verify that the file was transferred successfully. Use the **ls** command to list the contents of the home directory of the server and check for the file.



By mastering the **scp** command, you can securely transfer files between devices and protect sensitive data and systems from unauthorized access, eavesdropping, and tampering.

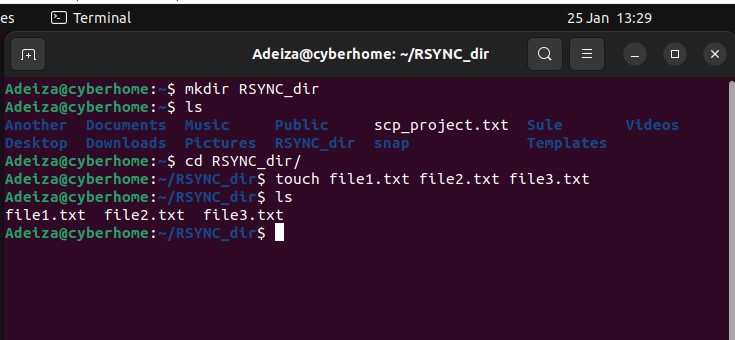
**TRANSFERING FILE USING RSYNC**

In this project, we will be creating a directory using the ‘**mkdir**’ command. Then we would make several files like say 3 different files inside the directory we made using the ‘**touch**’ command and send the file to the remote server using the **rsync.**

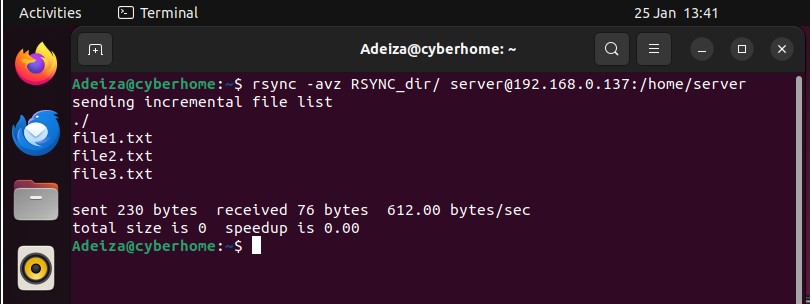
Like earlier, we would log in into the **server** **machine** using **ssh** protocol and use the ‘**rm**’ command to remove our initial file (scp\_project.txt) and also confirm the directory we sent is in the **server machine**.

We continue by exploring how powerful **rsync** is by logging back to our machine, use the ‘**mv**’ command to move the scp\_project.txt file inside the directory we initially sent, then use **rsync** to send it again. This is just to show that rsync will send files the first time and sync the file next time you are sending such file again instead of sending the whole file over again. Example, if we send a 40kb file the first time and changes were made to those files and it is now 60kb, rsync will send the 20kb extra to sync the initial one with the new one instead of sending the whole 60kb the second time.

1. Create a directory using the mkdir command. Example say we want to create a directory called “RSYNC\_dir” on our machine, we use the command $mkdir RSYNC\_dir. This will create a subdirectory or child-directory called RSYNC\_dir in our current root directory.
2. Create 3 .txt files inside the RSYNC\_dir directory we just created using the ‘touch’ command. Example, we want to create 3 .txt files named file\_1.txt, file\_2.txt, file\_3.txt, we will use the command:



1. Use Rsync to transfer the directory into the server machine while preserving its attribute (permission, ownership and timestamp e.t.c).

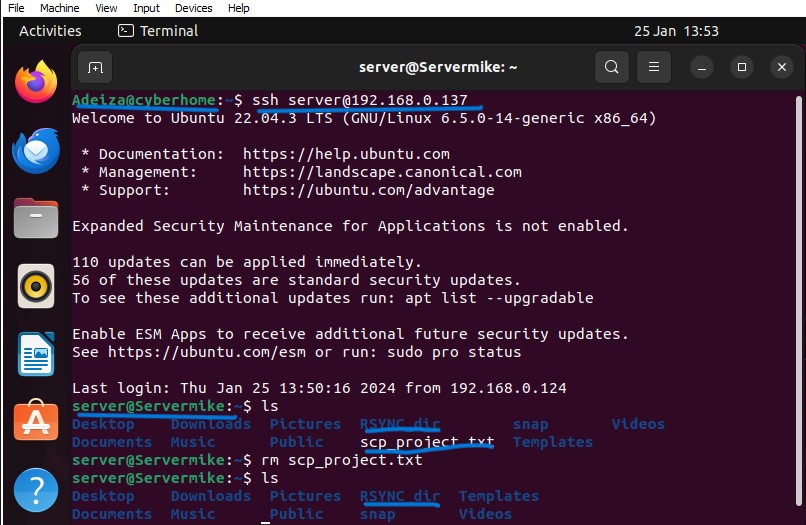


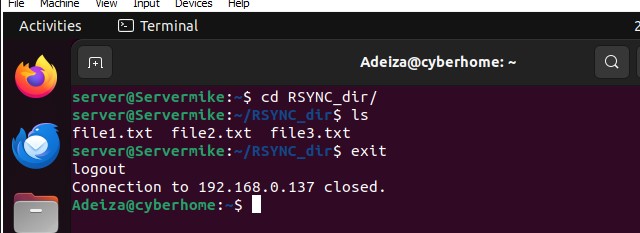
The **-avz** options used in the command **rsync -avz source\_directory/ user@server\_ip:/path/to/destination\_directory** are used to specify the desired behavior of the rsync command. Here's a breakdown of these options:

* **-a**: This option stands for "archive mode". It tells rsync to preserve file attributes and recursively transfer directories.
* **-v**: This option stands for "verbose". It tells rsync to display detailed information about the transfer process.
* **-z**: This option stands for "compress". It tells rsync to compress the data during the transfer to save bandwidth.

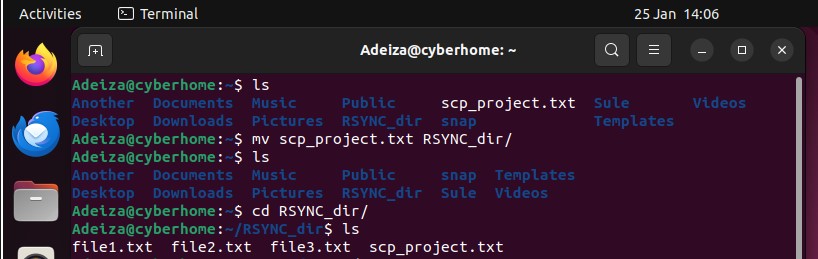
By using these options, we can efficiently transfer a directory and its contents to a server machine while preserving its attributes. Take note of the files sent from the directory, the bandwidth and bytes of data.

Login into the Server machine using **SSH,** confirm if our directory is there and remove our former text file (scp\_project.txt). To login into the server machine with the command **$ ssh user@remote\_ip** , then we use ‘**ls**’ to list homedir content. Use ‘**rm**’ command to remove the text file then confirm again and use ‘**exit’** command to log out of the server machine.

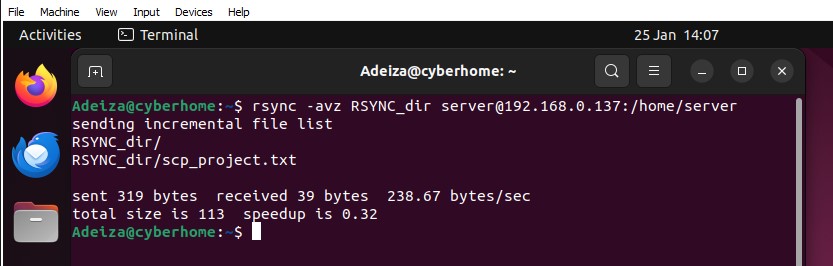




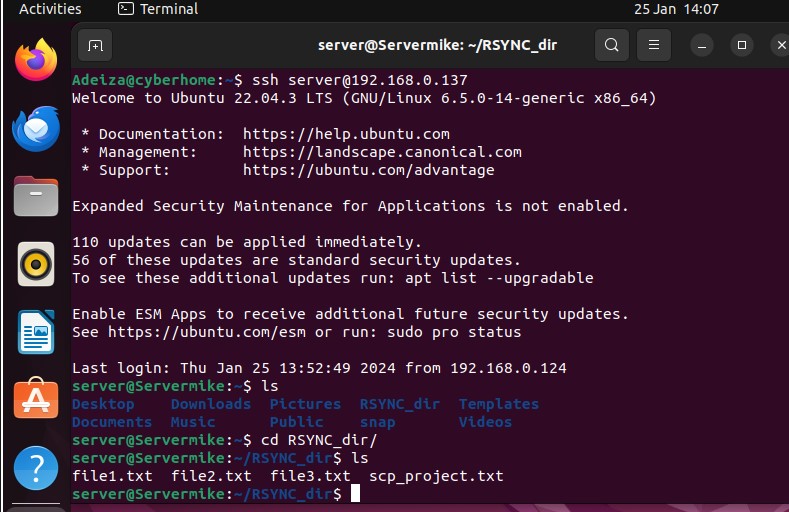
Now that we are logged out and back to our local machine, let’s use the ‘mv’ command to move the scp\_project.txt file into the RSYNC\_dir and transfer the directory again using Rsync. To use the ‘mv’ command in this case; $ mv scp\_project.txt RSYNC/. Here we use mv command to specify the file we want to move and to what location. The ‘mv’ command can also be used to rename a file.



Now that we moved the scp\_project file into the RSYNC\_dir, we need to send it to our remote machine using Rsync. Remember we sent it the first time and resending it again, Rsync will send only the scp\_project file while synchronizing the rest of the file we sent initially. To use Rsync like we did the last time we use the command: $ rsync –avz “directoryname” “user”@“remote\_ipaddr”:/location



Rsync only sent scp\_project file this time just like we discussed earlier. Now let’s login again using ssh and confirm from our server machine. Remember what we used to do that? Yes**! $ ssh user@remote\_ip**, use **$ ls** to list directory content and **$ cd “directory\_name”** to change to a directory.



In conclusion, SCP and Rsync are two powerful tools that can be used to transfer files between systems in a secure and efficient manner. While SCP is a simple and straightforward command that can be used to quickly transfer files, Rsync offers more advanced features such as delta encoding and compression, which can significantly reduce the time and resources required to transfer large files or directories.

Throughout the course of this project, I have explored both SCP and Rsync in-depth, gaining a deeper understanding of their respective features, advantages, and limitations.