

# Scotti-BYTE Enterprise Consulting Services

## Networking Tutorial: Accessing Remote Systems

There are many different ways in which to communicate between systems over the network. Some interfaces entail connecting a remote file system over the network and some involve connecting to a remote system to run a command.

The Secure Shell "ssh" is an encrypted protocol used to administer and communicate with remote servers. Desktop computers normally have a GUI Interface. Ubuntu servers normally only have a command line interface. Therefore, administering an Ubuntu server is likely done over the network from a terminal and "ssh" is a very common means to do so.

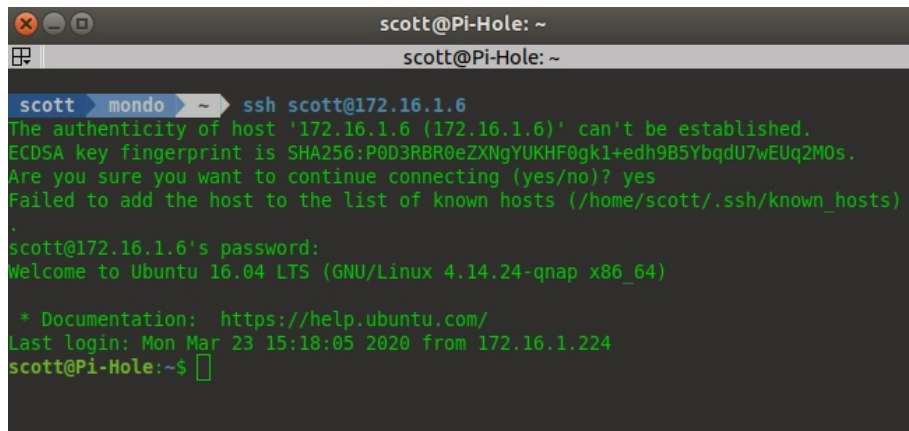
By default, "ssh" logs an identity or fingerprint for a remote system and if that changes, ssh assumes that the remote system might have been spoofed and does not immediately connect.

Also, by default, "ssh" accepts a username and password to log into the remote system.

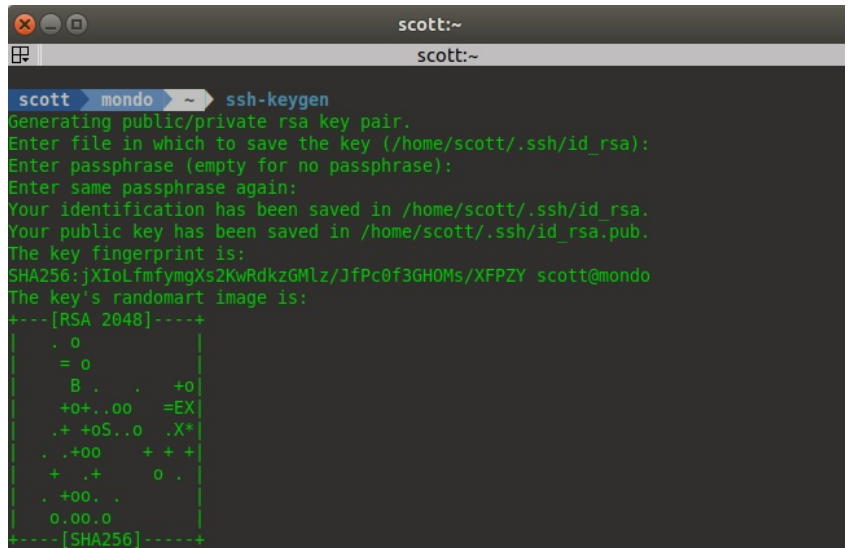
To increase security, you can establish an "RSA" crypto-key to verify and grant access to a system via the "ssh" protocol.

The first step is to create a key pair on your client machine. The client is the computer that will be used to connect to the remote server.

Use the "ssh-keygen" command. By default ssh-keygen will create a 2048-bit RSA key pair, which is secure enough for most use cases (you may optionally pass in the -b 4096 flag to create a larger 4096-bit key). The option to name the key exists if you have many keys that you want to be unique. Optionally, you can enter a



```
scott@Pi-Hole: ~  
scott@Pi-Hole: ~  
scott mondo ~ ssh scott@172.16.1.6  
The authenticity of host '172.16.1.6 (172.16.1.6)' can't be established.  
ECDSA key fingerprint is SHA256:P0D3RBR0eZXNgYUKHF0gk1+edh9B5YbqdU7wEUq2M0s.  
Are you sure you want to continue connecting (yes/no)? yes  
Failed to add the host to the list of known hosts (/home/scott/.ssh/known_hosts)  
.  
scott@172.16.1.6's password:  
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.14.24-qnap x86_64)  
  
* Documentation: https://help.ubuntu.com/  
Last login: Mon Mar 23 15:18:05 2020 from 172.16.1.224  
scott@Pi-Hole:~$
```



```
scott:~  
scott:~  
scott mondo ~ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/scott/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/scott/.ssh/id_rsa.  
Your public key has been saved in /home/scott/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:jXi0LfmfygmXs2KwRdkzGMLz/JfPc0f3GH0Ms/XFPZY scott@mondo  
The key's randomart image is:  
----[RSA 2048]-----  
 . 0  
  = 0  
  B . . +0]  
+0+...00 =EX]  
 .+ +0S...0 .X*]  
 .+00 + + +]  
 + .+ 0 .]  
 .+00 .]  
 0.00.0  
-----[SHA256]-----
```

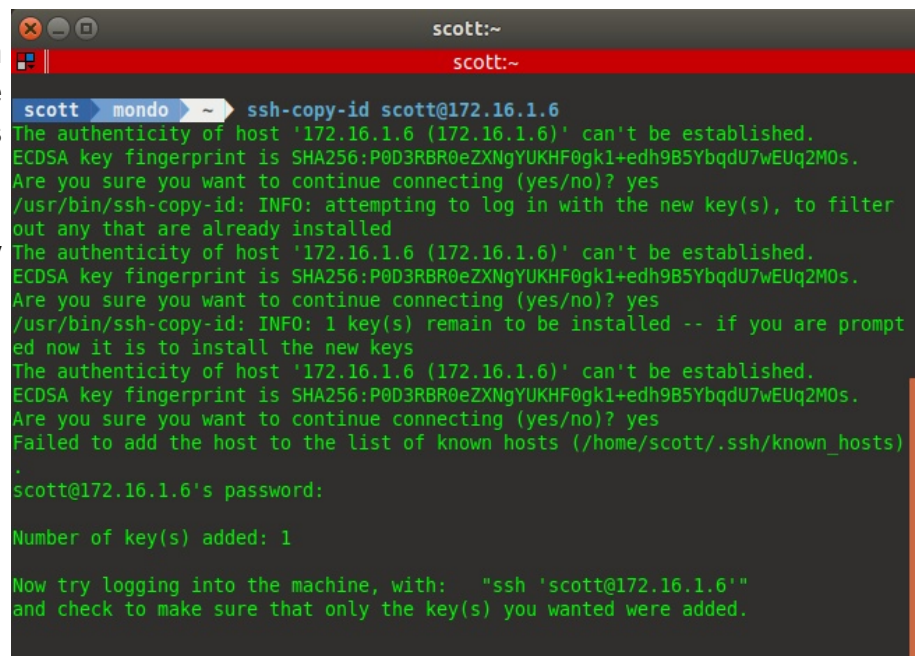
passphrase as well which provides another password challenge. Since we are using keys that will be unique to this username and system, I did not enter a passphrase.

At this point, you have a public and private key that you can use to authenticate. The next step is to place the public key on the remote system so that you can use SSH-key based authentication instead of a username/password to sign on to the remote system.

The quickest way to copy your public key to the Ubuntu host is to use a utility called ssh-copy-id.

Enter the command in "blue" in the screen image. You will be asked a couple times if this is what you want to do.

As you can see, the key is finally added.



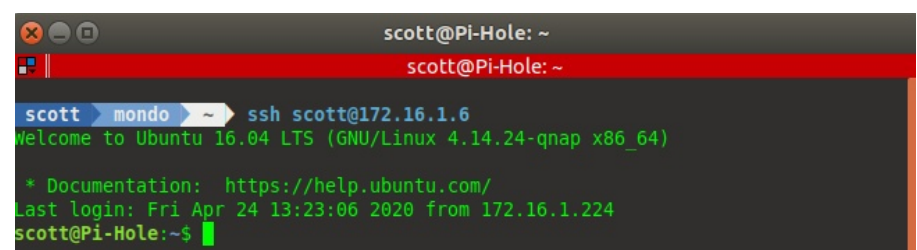
```
scott@mondo ~$ ssh-copy-id scott@172.16.1.6
The authenticity of host '172.16.1.6 (172.16.1.6)' can't be established.
ECDSA key fingerprint is SHA256:P0D3RBR0eZXNgYUKHF0gk1+edh9B5YbqdU7wEUq2M0s.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
The authenticity of host '172.16.1.6 (172.16.1.6)' can't be established.
ECDSA key fingerprint is SHA256:P0D3RBR0eZXNgYUKHF0gk1+edh9B5YbqdU7wEUq2M0s.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
The authenticity of host '172.16.1.6 (172.16.1.6)' can't be established.
ECDSA key fingerprint is SHA256:P0D3RBR0eZXNgYUKHF0gk1+edh9B5YbqdU7wEUq2M0s.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/scott/.ssh/known_hosts)
.
scott@172.16.1.6's password:
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'scott@172.16.1.6'"
and check to make sure that only the key(s) you wanted were added.
```

The screen at right shows that an "ssh" login now no longer asks for the password, which means the key is being used.

If you get the error "Failed to add the host to the list of known hosts", then you need to exit back to the system from which you initiated the "ssh" from and perform the following:

```
cd .ssh
sudo chown <username> known_hosts
```

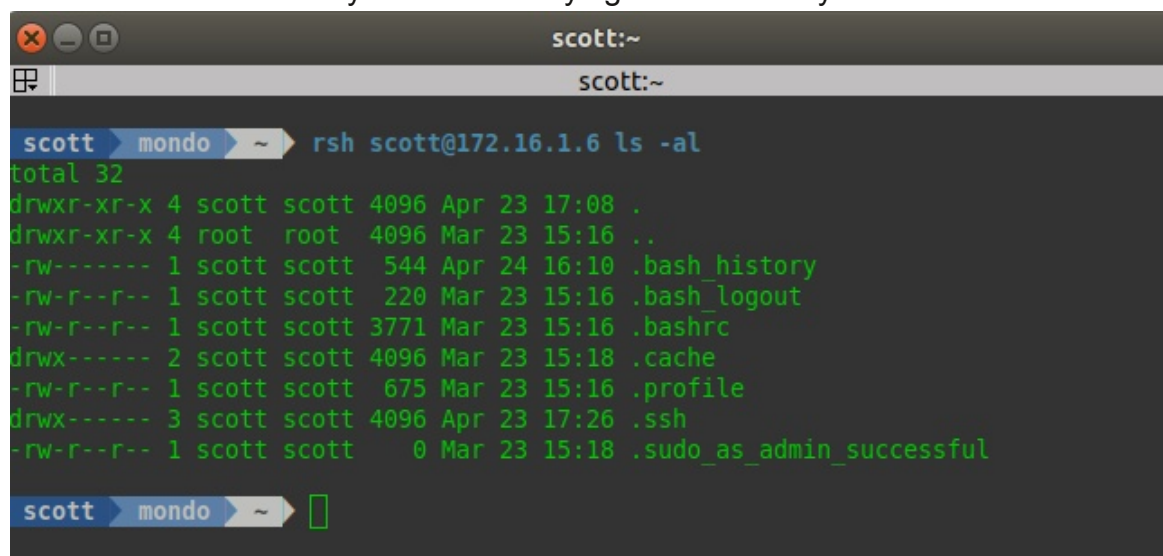
The next time that you log into the remote system, it will work without any messages as you can see in the screenshot at right.



```
scott@PI-Hole: ~$ ssh scott@172.16.1.6
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.14.24-qnap x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Apr 24 13:23:06 2020 from 172.16.1.224
scott@PI-Hole:~$
```

There is another command called remote shell "rsh" which is designed to execute a single command on a remote system while staying on the local system:

A terminal window titled 'scott:~' showing a command prompt 'scott@mondo'. The user enters 'rsh scott@172.16.1.6 ls -al'. The output shows a directory listing for the remote system, including files like .bash\_history, .bash\_logout, .bashrc, .cache, .profile, .ssh, and .sudo\_as\_admin\_successful.

```
scott@mondo ~$ rsh scott@172.16.1.6 ls -al
total 32
drwxr-xr-x 4 scott scott 4096 Apr 23 17:08 .
drwxr-xr-x 4 root  root  4096 Mar 23 15:16 ..
-rw-r--r-- 1 scott scott  544 Apr 24 16:10 .bash_history
-rw-r--r-- 1 scott scott  220 Mar 23 15:16 .bash_logout
-rw-r--r-- 1 scott scott 3771 Mar 23 15:16 .bashrc
drwx----- 2 scott scott 4096 Mar 23 15:18 .cache
-rw-r--r-- 1 scott scott  675 Mar 23 15:16 .profile
drwx----- 3 scott scott 4096 Apr 23 17:26 .ssh
-rw-r--r-- 1 scott scott    0 Mar 23 15:18 .sudo_as_admin_successful

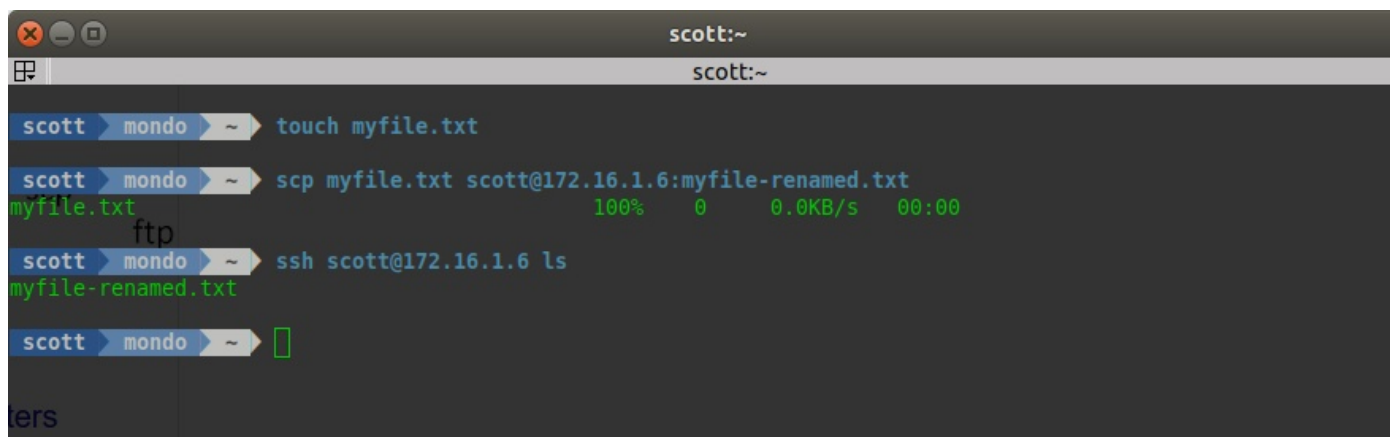
scott@mondo ~$
```

No login information was requested in the above command because of the RSA key that we shared in a previous step. In general, ssh is secure and encrypted, and rsh is not. The "ssh" command is a replacement for the older "telnet", "rlogin" and "rsh" commands. As you can see, the older commands do still exist.

There is also an "rexec" command and like "rsh" it allows you to execute a command on a remote system. The "rexec" command requires a password and is also insecure.

The "rcp" command is a "remote copy" and allows you to transfer files to and from another system over the network. Just like the above commands, there is an "scp" command which stands for secure copy and is recommended over "rcp".

In the following example, the "touch" command is creating a new file. The "scp" is secure copying the file to the remote system and renaming the file on the remote system during the copy. Finally, we do a secure shell "ssh" to show the directory listing on the remote system.

A terminal window titled 'scott:~' showing a sequence of commands: 'touch myfile.txt', 'scp myfile.txt scott@172.16.1.6:myfile-renamed.txt', and 'ssh scott@172.16.1.6 ls'. The output shows the file being created, copied successfully, and then listed on the remote system.

```
scott@mondo ~$ touch myfile.txt
scott@mondo ~$ scp myfile.txt scott@172.16.1.6:myfile-renamed.txt
myfile.txt                                100%  0  0.0KB/s  00:00
scott@mondo ~$ ssh scott@172.16.1.6 ls
myfile-renamed.txt

scott@mondo ~$
```

Another "ssh" command is a utility called the secure shell file system (SSHFS). This provides a way to actually mount a folder and file structure on a remote system thus making it appear to be part of the local file system. This "SSHFS" protocol is a file system client that uses the secure SSH File Transfer Protocol (sftp).



To install "sshfs":

```
sudo apt install sshfs
```

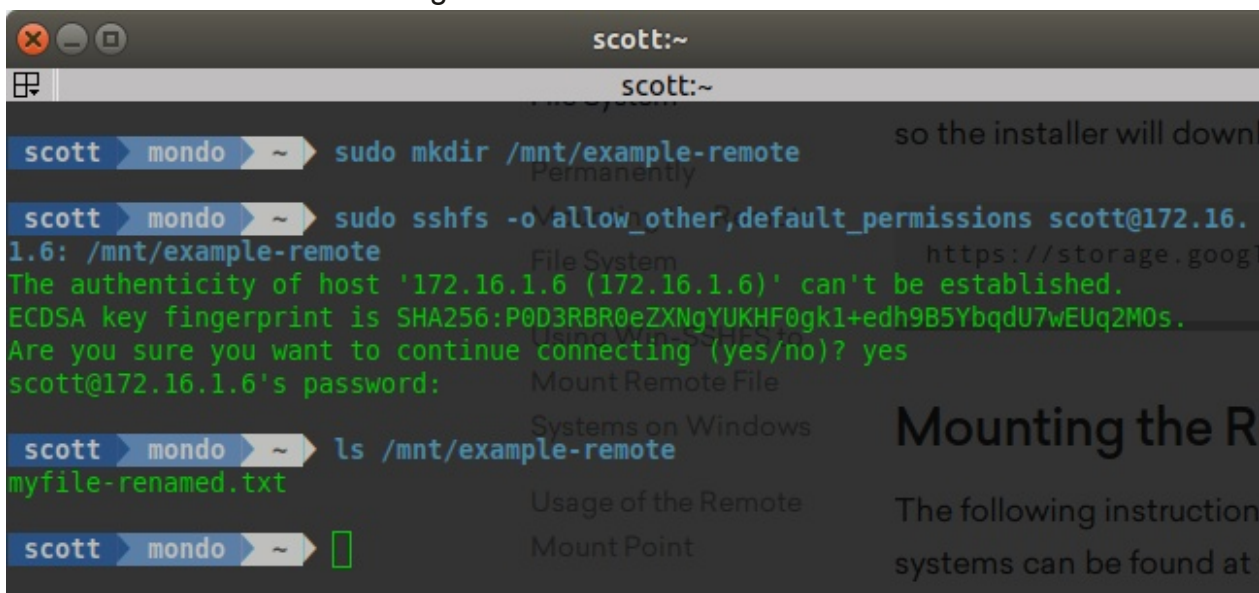
To mount a remote file system, you must first create a local directory in which to mount the remote system:

```
sudo mkdir /mnt/example-remote
```

Now we can mount a remote file system for local access:

```
sudo sshfs -o allow_other,default_permissions scott@172.16.1.6: /mnt/example-remote
```

You can see this in the following screenshot:

A terminal window titled 'scott:~' showing a series of commands and their outputs. The user 'scott' is logged into a machine named 'mondo'. The commands executed are: 1. 'sudo mkdir /mnt/example-remote' which succeeds. 2. 'sudo sshfs -o allow\_other,default\_permissions scott@172.16.1.6: /mnt/example-remote' which prompts for host authenticity, the user confirms 'yes', and asks for the password. 3. 'ls /mnt/example-remote' which lists 'myfile-renamed.txt'. The terminal output is color-coded: blue for prompts, green for success messages, and red for warnings or errors. The background of the terminal window shows a blurred view of a website with the title 'Mounting the R'.

Note that the "ls" command lists the file on our remote system through the local directory /mnt/example-remote. Any files added either locally to that folder or remotely will appear on both sides. Essentially, this is a simple and secure remote file system.

The "fusermount" is the command to unmount an "sshfs" mount. Normally "umount" can unmount most file systems, but "sshfs" is different.

File system in Userspace (FUSE) is a software interface that allows non-privileged users to create their own file systems. The only reason that our "sshfs" file system above used "sudo" was that we were mounted into "/mnt/example-remote" and the "/mnt" is a system area.

This mount could have just as easily been to a folder that was owned by a user. It is generally good practice to mount all file systems in "/mnt" if they are intended to be accessed by other users.

```
scott:~  
scott:~  
scott > mondo > ~ > sudo mkdir /mnt/example-remote  
scott > mondo > ~ > sudo sshfs -o allow_other,default_permissions scott@172.16.1.6: /mnt/example-remote  
The authenticity of host '172.16.1.6 (172.16.1.6)' can't be established.  
ECDSA key fingerprint is SHA256:P0D3RBR0eZXNgYUKHF0gk1+edh9B5YbqdU7wEUq2M0s.  
Are you sure you want to continue connecting (yes/no)? yes  
scott@172.16.1.6's password:  
scott > mondo > ~ > ls /mnt/example-remote  
myfile-renamed.txt  
scott > mondo > ~ > sudo fusermount -u /mnt/example-remote  
scott > mondo > ~ > ls /mnt/example-remote  
scott > mondo > ~ > 
```

Our example continues in the screen image above. The "fusermount" command unmounts the "sshfs" file system. The "ls" command now shows no files because the remote system is no longer connected and "/mnt/example-remote" is just an empty directory.

There are many other ways to access remote file systems. Linux provides "nfs" which is "Network File System". The "nfs" file system works by installing the server, creating an export directory, assigning server access to selected clients through the export file and exporting the shared directory.

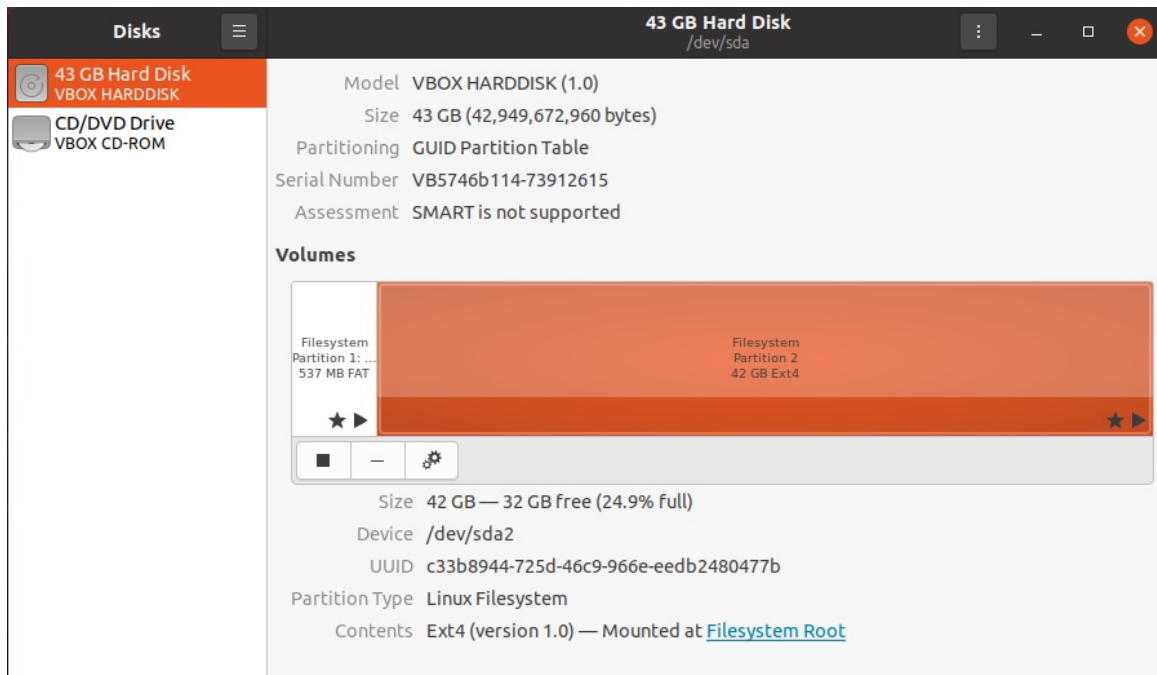
I have found that most linux variants usually install "Samba Server" which is a Windows Server Message Block/Common Internet File System (SMB/CIFS). Microsoft File Shares are so mainstream that I find it easier to use them from my Ubuntu system rather than NFS.

As long as you are connecting to and using file systems, you will find that it is critical to not only mount and unmount them dynamically, but mounting the file system to be statically available and persistent between reboots is desirable.

This paper has been written with remote file systems in mind. To mount a file system so that it is accessible between reboots, you need to edit the file system table file (/etc/fstab) to add a new mount entry.

The most common and mandatory entry you will see in the fstab is an entry that mounts the root (/) file system. The root file system is the core of the linux operating system and where everything starts. The ultimate God privilege in linux is called root (/) because anyone with root (/) privilege owns the file system. Everything in linux is a file and so "owning" root is the ultimate privilege.

To see how linux mounts the root file system in Ubuntu, launch the app called "disks" from the app drawer and highlight the linux file system. On an EFI booting GUID partition table, the system disk will always have a FAT32 EFI boot partition as the first partition and the second partition will be the root (/) partition containing the operating system.



Note the UUID in the above listing. When you list the `/etc/fstab` file, you will see the mount entry for the system partition which I highlighted below:

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options>      <dump> <pass>
# / was on /dev/sda2 during installation
UUID=c33b8944-725d-46c9-966e-eedb2480477b /          ext4    errors=remount-ro 0    1
# /boot/efi was on /dev/sda1 during installation
UUID=C8E2-D1A1 /boot/efi    vfat    umask=0077      0    1
```

You will also see the mount entry for the efi boot partition two lines down and you can reference that partition similarly in the "disks" app. If you had another hard disk physically installed in your system which you wanted to persistently mount, you would add it to the `/etc/fstab`.

On my system, I have a spinning drive for which I creating a `/mnt/MondoSeagate` mount folder for. I referenced the "disks" app to find its UUID and the mount command is as follows:

```
UUID=7a7c3efa-951a-4be2-9bcb-b8193a5f38ae /mnt/MondoSeagate ext4 rw,exec 0 0
```

To mount an SSHFS disk persistently in the `/etc/fstab`, the entry would look like:

```
username@host:/remote/dir /mnt/mount-folder fuse.sshfs defaults 0 0
```

In order to persistently mount a CIFS Windows share at boot time on an Ubuntu system, you will need to install a utility package:

```
sudo apt update
sudo apt install cifs-utils
```

First create the mount point directory:

```
sudo mkdir /mnt/win_share
```

To mount the share from the command line:

```
sudo mount -t cifs -o username=<windows-user> //server-ip/sharename
```

You will be prompted for the password.

In order to mount a CIFS share in the `/etc/fstab`, the entry might look like this:

```
//server-ip/sharename /mnt/sharename cifs
username=x,password=y,dir_mode=0777,file_mode=0777 0 0
```

The above command would be on a single line and would grant everyone on the Linux system full read and write access to the remote CIFS share. Usually, linux users might mount a CIFS share on a per user and per session basis from the GUI File manager.

Also, it is more secure to put the user credentials in a separate file only accessible by root (/).

However, there are times when a persistent mount through the `fstab` might be very desirable.

Mount commands in `/etc/fstab` have a huge number of options. I have provided only the most basic entries.

The `/etc/fstab` was mentioned above in the context of accessing remote systems. I've included this very basic information in hopes that it persistent mounts could be used for `sshfs`, `nfs`, and `cifs` files. The `/etc/fstab` is also the place to mount `iscsi` drives as well.

`Ischi` is a very high performance network based file system and is beyond the scope of this paper. Hopefully the "r-" services and their secure "s" variants described herein will be helpful tools in communicating between linux systems and even non-linux systems from linux on a network.