How to use SSH Pipe on Linux

The UNIX pipe is a major step forward in the development of UNIX and UNIX-like operating systems. It allows users to perform complex computational tasks by linking together the input and output of basic programs.

The UNIX pipe is a major step forward in the development of UNIX and UNIX-like operating systems. It allows users to perform complex computational tasks by linking together the input and output of basic programs. This article expands on that by showing you how to use UNIX pipes in Linux over the network using the SSH protocol.

Learn about Unix Pipeline

Pipes on Unix (and by extension Linux) are used to connect programs and make them work together. For example, when using cat, you can display the contents of the file, but if you use a pipe (|), you can chain the cat command with the more command to make reading the file easier.

```
cat file1 | more
```

The basic idea here is: program1 fileX | program2. However, it is not limited to just one file and two programs. Piping can become more advanced according to your needs, with as many modifiers as you can think of.

Here are some ways to make good use of pipe (|) in SSH situations.

Automatically convert compressed folders

One of the most common ways to use the UNIX pipe is to store the program's output to a file somewhere on the local system. For example, run echo "Hello, MakeTechEasier!" | tee Hello will run the echo program while also storing the string 'Hello, MakeTechEasier!' inside the file 'Hello'.

That means you can use this idea to transfer folders across two Linux servers. To do that, read the directory you want to send with tar, then pass it to the SSH daemon:

```
tar czf - "~/Documents/myfolder" | ssh ramces@remote.host "tar xzf - -C ~/Documer
```

This command will package your directory into a tar archive and send it to the command's standard output. The UNIX pipe will then read that data and send it to your remote Linux server using SSH.

You can also reverse this command to get your files off the remote server:

```
ssh ramces@remote.host "tar czf - ~/Documents/myfolder" | tar xzf - -C "~/Documer
```

Push and retrieve files from remote servers

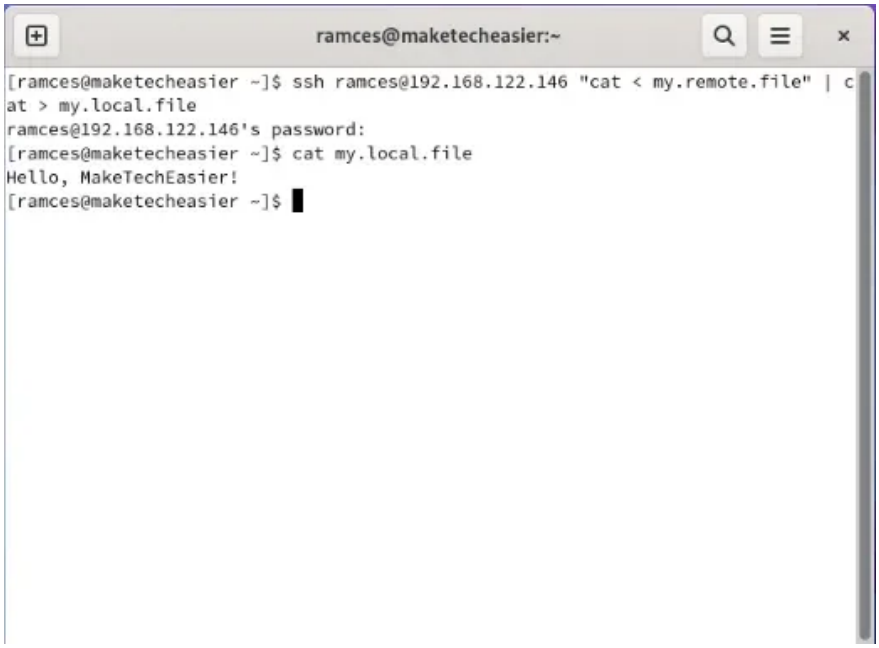
You can also use pipes and SSH to send individual files over the network. This works by using cat as a way to download the file's contents and send it over SSH:

```
cat  my.local.file | ssh ramces@remote.host "cat > my.remote.file"
```

The remote server will receive the output stream from the local cat process and rebuild the file as before.

To retrieve files from a remote server, you need to reverse the order of the commands and provide the path to the remote file:

```
ssh ramces@remote.host "cat  my.remote.file" | cat > my.local.file
```



Backup and restore remote drives

Similar to sending files and folders, it is possible to remotely back up entire drives in Linux using UNIX pipes and SSH. This can be useful if you want to create a quick offsite backup and you don't currently have a spare physical drive.

To backup an entire drive, run dd with the 'if=' variable set to the drive you want to backup, then pipe it to your SSH daemon:

```
sudo dd if=/dev/sda | ssh ramces@remote.host "dd of=sda.img"
```

Reversing this command also allows you to restore the disk image from the remote machine to the physical drive:

```
ssh ramces@remote.host "dd if=sda.img" | sudo dd of=/dev/sda
```

Furthermore, this SSH pipe syntax will also work with discrete drive partitions. For example, if your system has a /home partition in '/dev/sda4', you can run the following command to create a backup of that partition:

```
sudo dd if=/dev/sda4 | ssh ramces@remote.host "dd of=home.img"
```

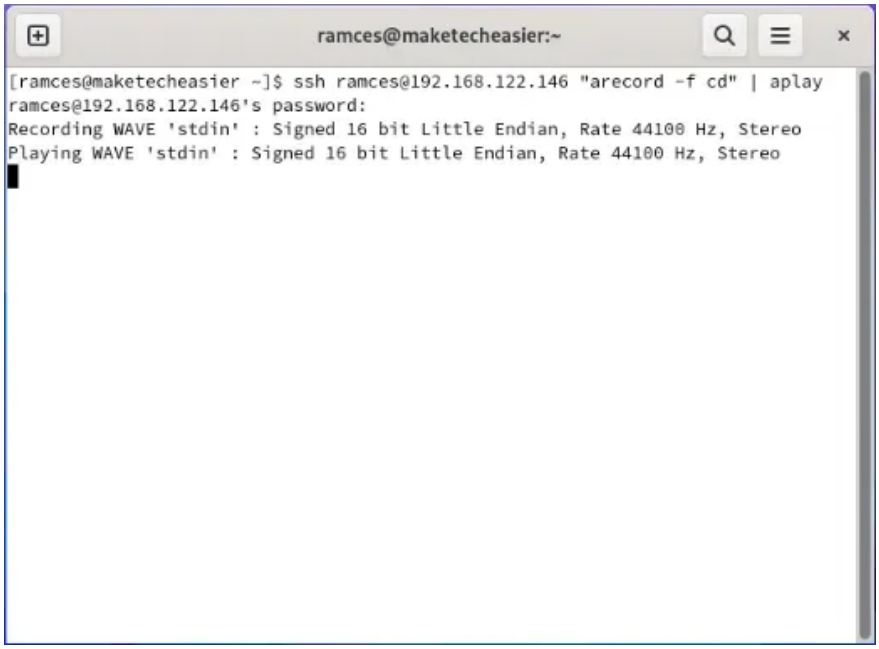
Redirect audio input to the remote machine

One of the benefits of SSH pipes is that they allow you to interact with remote machines as if they were local resources. This includes the ability to exploit device files, such as the system's audio input.

To do this, run a remote ALSA subshell using SSH and send its output to your local ALSA daemon:

```
ssh ramces@remote.host "arecord -f cd" | aplay
```

This will listen for the default audio input device on the remote machine and play what it hears on your system. That means, flipping the commands around sends the local machine's audio input to the remote server's audio output:




```
arecord -f cd | ssh ramces@remote.host "aplay"
```

ALSA SSH pipe will also work when you combine it with other audio streaming tools. For example, you can send arecord output from an SSH pipe to ffmpeg:

```
ssh ramces@remote.host "arecord -f cd" | ffplay -nodisp -
```

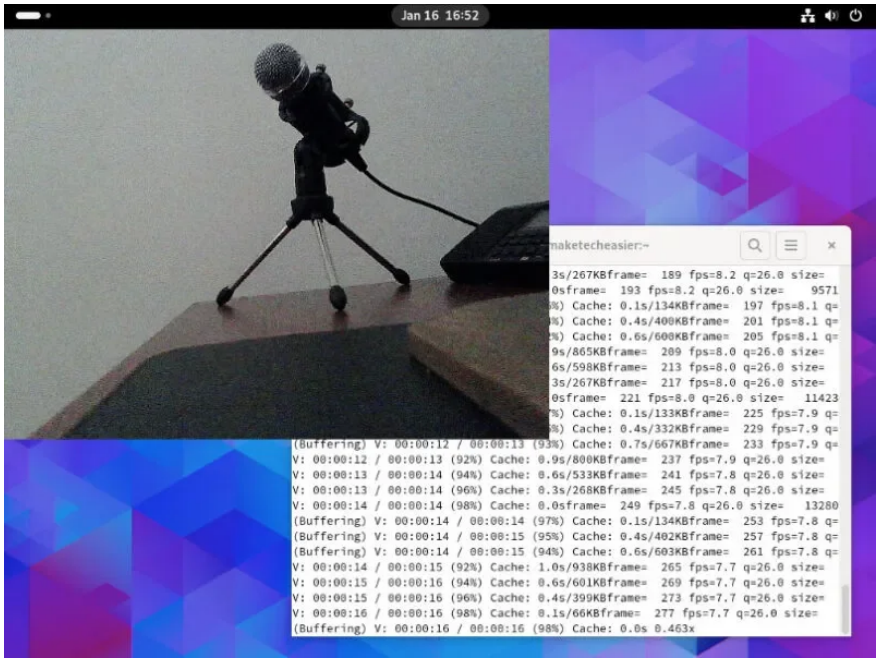
Stream live video from a remote webcam

Another great use of SSH pipes in Linux is to stream live video webcam feeds. Just like audio, this allows you to take advantage of a remote server's device and display that device's output on your local machine.

To stream from a remote server's webcam, run SSH with the ffmpeg subshell, then pass it to the video streaming client on your local machine:

```
ssh ramces@remote.host "ffmpeg -r 14 -s 640x480 -f video4linux2 -i /dev/video0 -
```

This command will stream raw video from the first webcam on your remote machine.



You can also record footage from a remote webcam to a separate file. You can do this by sending data from the SSH pipe to tee before redirecting it to your video player:

```
ssh ramces@remote.host "ffmpeg -r 14 -s 640x480 -f video4linux2 -i /dev/video0 -
```

Print text on remote console

In addition to audio and video, you can also use the SSH pipe to send raw text on the remote machine's TTY. This is useful if you want to send status messages to a system that does not have a GUI.

To get started, create a FIFO pipe on your local machine:

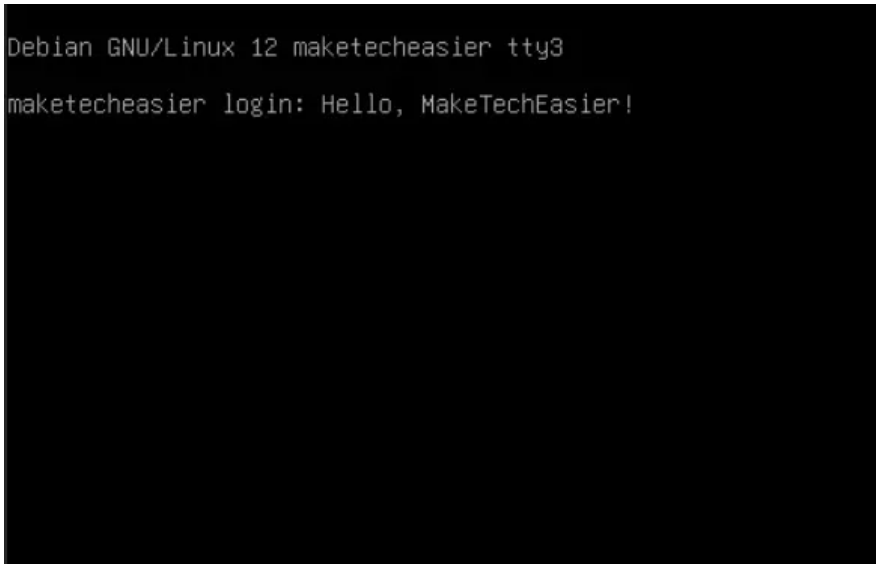
```
mkfifo my-fifo
```

Run the tail listen command using your FIFO and send its output to the SSH daemon:

```
tail -f my-fifo | ssh root@remote.host "cat > /dev/tty0"
```

Test that your new FIFO pipe works over the network by sending text data with the echo command:

```
echo "Hello, MakeTechEasier!" > my-fifo
```



Note: Sending text to a device's TTY will only work if you're logged in with that device's root account.

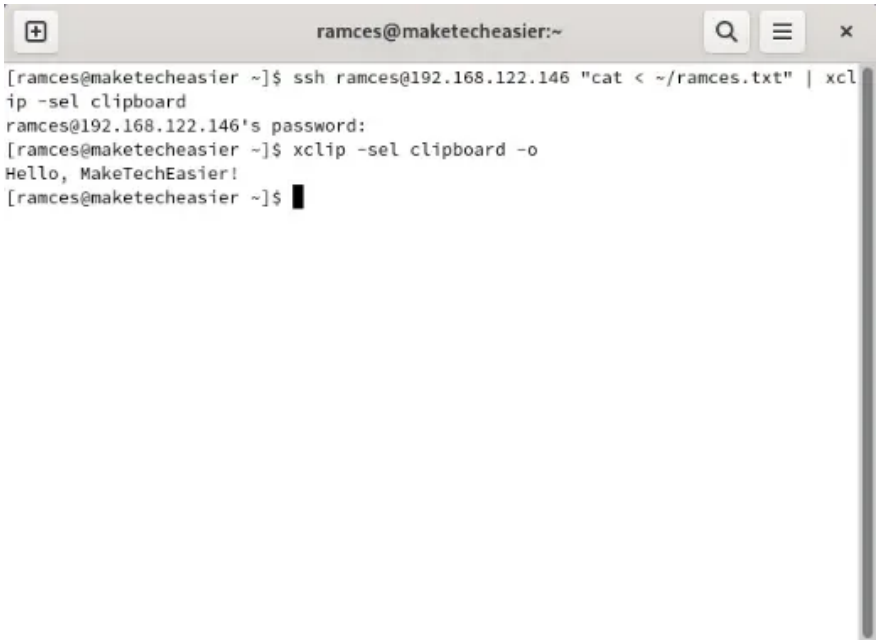
Bring remote data to local clipboard

The biggest disadvantage of the system clipboard is that it only works with the local machine. This is a problem if you are working with multiple computers and want to transfer data without creating temporary files.

One way to fix this is to create an SSH pipe that can read and send remote files directly to your local system clipboard:

```
ssh ramces@remote.host "cat  ~/ramces.txt" | xclip -sel clipboard
```

This command will connect to your remote machine, run the cat utility then start reading the 'ramces.txt' file. Once completed, it sends the remote data back to your local machine and redirects it to the system clipboard.



You can also push the system's current clipboard as a file on the remote machine by using the following command:

```
xclip -sel clipboard -o | ssh ramces@remote.host "cat > ~/clip.txt"
```

Learning how to send data over a network using UNIX pipes and SSH is just the first step in understanding how computer networks work. Learn more about your network by tracking where your packets go with Traceroute.