Using sshfs on C accessed via A and B

Ask Question

2

The current set up I am working with requires me to ssh onto a machine C by first ssh-ing into A and then into B. Something like this:

```
ssh [username]@[university].com # Machine A
ssh [department] # Machine B
ssh [machinename] # Machine C
```

I am asked for a password when ssh-ing into A.

My goal is to use sshfs to mount a remote directory on C locally in order to edit files in the remote directory using file manager and text editors locally on my machine. I have tried to follow multiple tutorials/blog posts but cannot seem to get this working.

I have been trying to add entries to my `~/.ssh/config` file to automate the entire ssh-ing process first, but to no avail. Can someone please explain how I can do this or provide links that can help?

As a side question: is the intended approach the easiest for my purposes (i.e. being able to edit/create scripts on a remote drive using local text editors)?

I am on Ubuntu 18.04 and am quite new to Linux. Thank you for your help!

ssh remote-access sshfs Edit tags

share edit close flag

asked Sep 11 '18 at 12:43
dstivd
11 1

Slightly different problem but maybe the solution at askubuntu.com/questions/980883/… can guide you? – Organic Marble Sep 11 '18 at 13:06

I'm agree with @OrganicMarble, you should create tunnel, then you will be able to connect from A to C through the tunnel created on B. Here is another example: askubuntu.com/q/1005337/566421. – pa4080 Sep 11 '18 at 17:45

add a comment

start a bounty

1 Answer

active oldest votes

0

I'm assuming you are working on *machine A*. From there you can connect to *machine B*. And from *B* you can connect to *machine C*. That you want is to establish a connection *directly* from *machine A* to *machine C*. Here is my suggestion:

- Create SSH connection from *machine B* to *machine C* and forward a local port on *B* (`20022` - for example) to the remote SSH port on *C* (`22`).

  ```
  machine-b~$ ssh machine-c -L 20022:127.0.0.1:22 -fTN
  ```

  Or issue the command through SSH directly from *machine A*:

  ```
  machine-a~$ ssh machine-b "ssh machine-c -L 20022:127.0.0.1:22 -fTN"
  ```

- Create SSH connection from *machine A* to *machine B* and forward a local port on *A* (`10022`) to the remote port `20022` on *B*.

  ```
  machine-a~$ ssh machine-b -L 10022:127.0.0.1:20022 -fTN
  ```

- Create SSH connection or use SSHFS from *A* to its loopback iface (`127.0.0.0` / `localhost`) on port `10022` to connect to *C*.

  ```
  machine-a~$ ssh -p 10022 <machine-c-user>@localhost
  machine-a~$ sshfs -p 10022 <machine-c-user>@localhost:/target/dir/ /dest/dir/
  ```

According to this scenario below are presented few little bit more advanced ideas and explanations.

1. Setup *machine B* to connect to *machine C* by using ssh key pair (without passphrase), if this is not previously done. For this purpose use the following sequence:

- SSH to *machine B* and generating RSA key pair, don't enter passphrase:

  ```
  mkdir -p ~/.ssh/machine-c-key/
  chmod 700 ~/.ssh
  chmod 700 ~/.ssh/machine-c-key/
  ```

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/machine-c-key/id_rsa   # do not enter passphrase
chmod 600 ~/.ssh/machine-c-key/id_rsa
```

- Transfer the `id_rsa.pub` key into the `~/.ssh/`**`authorized_keys`** file on *machine C*, by using `ssh-cipy-id`. On *machine B* execute the command:

```
ssh-copy-id -i ~/.ssh/machine-c-key/id_rsa '<user>@<machine-c>' -p '22'
```

- Now try logging into the *machine C*, with:

```
ssh -i ~/.ssh/machine-c-key/id_rsa -p '22' '<user>@<machine-c>'
```

  If this works, if you wish, edit `/etc/ssh/sshd_config` and set: `PasswordAuthentication no`

- Copy the generated key from *machine B* to *machine A*, we will use it later. On *machine A* use `rsync` in the following way (pay attention to the slashes `/`):

```
rsync -r machine-b:/home/<user>/.ssh/machine-c-key ~/.ssh/
```

  Deal with the permissions (on *machine A*):

```
chown -R $(id -u):$(id -g) ~/.ssh
chmod 700 ~/.ssh/machine-c-key/
chmod 600 ~/.ssh/machine-c-key/id_rsa
```

2. Create `~/.ssh/`**`config`** file on *machine B*, and setup port forwarding. Add these lines into the mentioned config file:

```
Host machine-c-port-fwd
        HostName 192.168.100.100
        IdentityFile ~/.ssh/machine-c-key/id_rsa
        User user
        Port 22
        LocalForward 20022 127.0.0.1:22
```

- You should provide the actual *IP-address or domain name* of *machine C* and the actual *user*.
- The `Port 22` is the ssh port on *machine C*.
- The directive `RemoteForward 20022 127.0.0.1:22` means that the port `22` of remote loopback interface (on *machine C*) will be forwarded to port `20022` on *machine B*.

- Note if you already have configuration for `Host machine-c` it is a good idea to create a separate one for the port forwarding setup.

Now you should be able to connect to *machine C* from *machine B* by using of the command:

```
ssh machine-c-port-fwd
```

We can push this connection into the background by adding the options `-fTN` (reference). We could use also the tool `autossh` to be sure the connection will be kept alive for a long period of time (`sudo apt install autossh` - reference):

```
autossh machine-c-port-fwd -fTN
```

- Use `killall autossh` to disable it :)

At this stage you should be able to connect to *machine C* from *machine B* through its loopback interface:

```
ssh -i ~/.ssh/machine-c-key/id_rsa -p '20022' '<machine-c-user>@localhost'
```

3. Establish connection from *machine A* to *machine C* (through the *machine B*). Perform the following steps on *machine A*:

- Issue a command via SSH to *machine B* to establish a connection to *machine C*:

  ```
  ssh machine-b "autossh machine-c-port-fwd -fTN"
  ```

  - Yo can add the following job in the user's Crontab (`crontab -e`) on the *machine B* to establish the port forwarding on system reboot:

    ```
    @reboot sleep 15 && autossh machine-c-port-fwd -fTN
    ```

- Establish SSH connection to *machine B* (from *A*) with port forwarding and push the connection into the background:

  ```
  autossh machine-b -L 10022:127.0.0.1:20022 -fTN
  ```

  - In this way the local port `10022` (on *machine A*) will be forwarded (bind) to the remote port `20022` (on *machine B*) that is bind to port `22` on *machine C*. We can use the same port number (`20022` - for example) on *A* and *B*...

  - Note that everywhere we are limit the port forwarding only to the loopback interface (`127.0.0.1`), but this could be done for all interfaces (`0.0.0.0` or `\*`) or for certain interface (by using its IP).
  - You can create `~/.ssh/config` file on *machine A*, analogical to the description in section 2, to simplify your commands in the future. Alco I would prefer to use SSH key based authentication...
- Connect to *machine C* from *machine A* through its (on *machine A*) loopback interface:

```
ssh -i ~/.ssh/machine-c-key/id_rsa -p '10022' '<machine-c-user>@localhost'
```

  Or use SSHFS in this way (note, `sshfs` can use the hosts from `~/.ssh/config`):

```
sshfs -p 10022 <machine-c-user>@localhost:/target/dir/ /destination/dir/ -o IdentityFile=~/
```

To automate the execution of the above three commands you can create a function as the follow in your `~/.bashrc` file:

```
mount-machine-c() {
    ssh machine-b "autossh machine-c-port-fwd -fTN" && sleep 1
    autossh machine-b -L 10022:127.0.0.1:20022 -fTN
    sshfs -p 10022 <machine-c-user>@localhost:/target/dir/ /destination/dir/ -o IdentityFile=~/
}
```

`source ~/.bashrc` and `mount-machine-c` will be available as shell command. You can add additional logic and arguments to test whether `/destination/dir/` is already mounted, etc...

share edit delete flag

edited Sep 12 '18 at 6:48

answered Sep 11 '18 at 22:27

pa4080
13.9k 5 25 64

add a comment