Dig a multi-hop ssh tunnel

Suppose you can not reach a server directly with ssh, but only via multiple intermediate hosts (for example because of routing issues). Sometimes it is still necessary to get a direct client - server connection, for example to copy files with scp, or forward other ports like smb or vnc. One way to do this is to chain tunnels together to forward a port to the server along the hops. This "carrier" port only reaches its final destination on the last connection to the server.
Suppose we want to forward the ssh port from a client to a server over two hops. Once the tunnel is build, it is possible to connect to the server directly from the client (and also add an other port forward).

Create tunnel in one shell

client -> host1 -> host2 -> server and dig tunnel 5678

```
client>$ ssh -L5678:localhost:5678 host1        # 5678 is an arbitrary port for the tunnel
host_1>$ ssh -L5678:localhost:5678 host2        # chain 5678 from host1 to host2
host_2>$ ssh -L5678:localhost:22 server         # end the tunnel on port 22 on the server
```

Use tunnel with an other shell

client -> server using tunnel 5678

```
$ ssh -p 5678 localhost                         # connect directly from client to  server
$ scp -P 5678 myfile localhost:/tmp/            # or copy a file directly using the tunnel
$ rsync -e 'ssh -p 5678' myfile localhost:/tmp/ # or rsync a file directly to the server
```

Autoconnect and keep alive script

I use variations of the following script to keep a machine reacheable over a reverse ssh tunnel. The connection is automatically rebuilt if closed. You can add multiple `-L` or `-R` tunnels on one line.

```
#!/bin/sh
COMMAND="ssh -N -f -g -R 3022:localhost:22 colin@cb.vu"
pgrep -f -x "$COMMAND" > /dev/null 2>&1 || $COMMAND
exit 0
```

```
1 * * * * colin /home/colin/port_forward.sh     # crontab entry (here hourly)
```