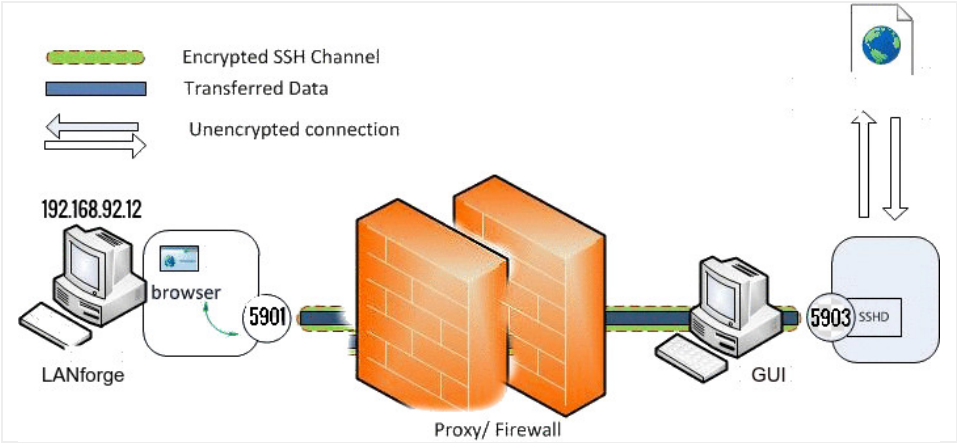
sales@candelatech.com
support@candelatech.com
+1 (360) 380-1618 [PST, GMT -8]

Set up an SSH-tunnel on Windows, Linux, or Mac

Goal: Connect to a LANforge Linux system via a compressed tunnel connection

When connecting to your remote LANforge hardware (presumably accessible over a VPN) you will notice poor response time and lag in your LANforge GUI or your VNC connection. Many VPN connections are based on UDP protocols and packet loss might be affecting your connection quality. Below we explain how to set up SSH tunnels that increase the quality of your connection.



1. ## Linux SSH Tunnel Setup

A. Forwarding a Single Port

A. The ssh option *-L* is takes an argument **local-port:remote-ip:remote-port**. The *remote-ip* parameter does not have to match the destination host (but it may). VNC display **:1** uses the port *5901*. When VNC is in **localhost** mode, it binds to *127.0.0.1:5901*.
The *local-port* parameter is the port on the local computer. It probably won't correspond to the remote port.
The resulting command looks like: `ssh -L 5900:localhost:5901 user@remotehost`.
When connecting a VNC browser to *localhost:0* (or *localhost::5900*) it will forward packets to **remotehost**, and the SSH service on **remotehost** will forward them to the *localhost:5901* port.If you are forwarding multiple LANforge VNC ports to your laptop, you will want to **make a plan** for what local ports you want to use.
Multiple remote VNC sessions would be forwarded using multiple ssh sessions:

```
$ ssh -CnNL 5901:localhost:5901 lanforge@ct523c-8a33
$ ssh -CnNL 5902:localhost:5901 lanforge@ct523c-fc30
$ ssh -CnNL 5903:localhost:5901 lanforge@ct521a-110b
$ ssh -CnNL 5904:localhost:5901 lanforge@ct523-3231
```

Using the above set of commands, you can connect your VNC viewers multiple X11 display ports on your laptop:

- localhost:1
- localhost:2
- localhost:3
- localhost:4

B. Other SSH Parameters

```
124 alias     FreyaTunnel="ssh -CnNv -L 5903:192.168.92.13:5901  -L 4131:192.168.92.13:4001   -L 4132:192.168.92.13:4002
    lanforge@192.168.92.13"
```

I. `-C`: Requests compression of data. This is desirable for slower connections. **Recommended**.

II. `-n`: redirects stdin from `/dev/null`. Required when SSH is running in the background.

III. `-N`: do not execute a remote command, useful when forwarding ports.

IV. `-v`: Verbose mode. Causes SSH to print debugging messages about its progress.

V. `-L local-ip:local-port:remote-host:remote-port`.
Use this flag multiple times to forward multiple ports with one command.

VI. Usually the `-L` forward uses three parameters, as seen above. Ask support if you need to forward a remote port to only one of your laptop network interfaces.

For more information see Please visit the SSH man page for further flags and switches

B. Multiple Forwards to One Host

A. SSH can support multiple port forward per remote host.
`ssh -L localport:ipaddress:remoteport user@remotehost`.
Below are ports that you probably want to forward:

I. `4001` -- perl scripts use this for ascii connection to LANforge server

II. `4002` -- GUI uses this for binary connection to LANforge server

III. `5901` -- VNC port for display :1

IV. `8080` -- REST API port provided by remote GUI

B. These can be combined into multiple command line arguments. The example below forwards all LANforge ports to your laptop:

```
$ ssh -CnNv -L 4001:localhost:4001 \
-L 4002:localhost:4002 \
-L 5900:localhost:5901 \
user@192.168.100.1
```

Notice that in a secure VNC and secured LANforge configuration, this will forward the remote hosts localhost bound ports to your laptop.

C. Indirect Host Access

A. Your laptop might **not have direct ssh access to the LANforge machine** Instead, you might have ssh access a *gateway* or *jump host* machine that is a firewall between the LANforge and your laptop. This can present itself in two ways:
a) you can ssh to the jump host, but not beyond it
b) you cannot ssh to the firewall, but it provides port forwards for LANforge services

B. You can ssh to a jump host

I. You still need to know what the remote LANforge IP is.

II. Your ssh command would look like:

III. `ssh -CnN -L4001:lanforgeip:4001 user@jumphost`

C. You *cannot ssh* to the firewall

In this case, *ssh will not be useful.* You will have to point the GUI or python script on your laptop to the remote port on the firewall.

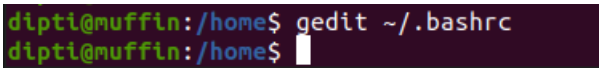
I. The firewall forwards port 34002 to `lanforge-1:4002`

II. Connect your GUI to `firewall:34002`

III. Your firewall administrator will need to share the port forwards on the firewall.

D. Updating your shell aliases

A. From the computer that you are trying to connect your SSH tunnel from, open the `.bashrc` file from `/home/user/`. The `.bashrc` file can be opened via *gedit*, *vim*, or *nano*. This `.bashrc` file is where the alias will be setup to properly invoke your ssh.



B. Once the `.bashrc` file is open, type in your alias in any blank spot (that is not within another for-loop or definition).

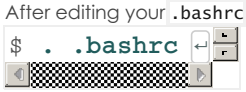
C. Further example ssh aliases include:

```
124 alias    FreyaTunnel="ssh -CnNv -L 5903:192.168.92.13:5901  -L 4131:192.168.92.13:4001   -L 4132:192.168.92.13:4002
    lanforge@192.168.92.13"
```

I.
```
alias FreyaTunnel="ssh -CnNv -L 5903:192.168.0.6:5901 \
-L 4131:192.168.0.6:4001 \
-L 4132:192.168.0.6:4002 \
lanforge@192.168.0.6"
```

II. `alias SaltTunnel="ssh -CnN -L 4001:192.168.200.18:4001 salt@10.253.1.6"`

D. After editing your `.bashrc` file, source the file to apply the changes:



E. In order for our machine to remember certain passwords and access configurations, some additional edits in the ssh config file.This will be in your ,,~/.ssh/config `file (or` $HOME/.ssh/config,,).

E. **SSH Keys**

A. The ssh connection might require an ssh key. This means that one needs to be generated. The private key and public key of the key pair must be saved to the local computer. The public key of the pair should be copied to the remote computer.

B. Add your SSH key to the device being forwarded. Finally, add your public key that you generated earlier via SSH. This can be done by typing in `ssh-copy-id user@ipaddress` (see below example).

```
dipti@muffin:~$ ssh-copy-id lanforge@192.168.95.101
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/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
lanforge@192.168.95.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'lanforge@192.168.95.101'"
and check to make sure that only the key(s) you wanted were added.
```

C. Once the alias is added to .bashrc file and the ssh key is added to the remote device, open any terminal and simply type in the alias name. This will initiate the tunnel. For example, "FreyaTunnel" in this example would be the alias typed into any terminal. This should incur an instance of your tunnel.

2. # Windows SSH tunnel Setup

A. There are many ways to set up an SSH tunnel, however, this cookbook will utilize PuTTy.

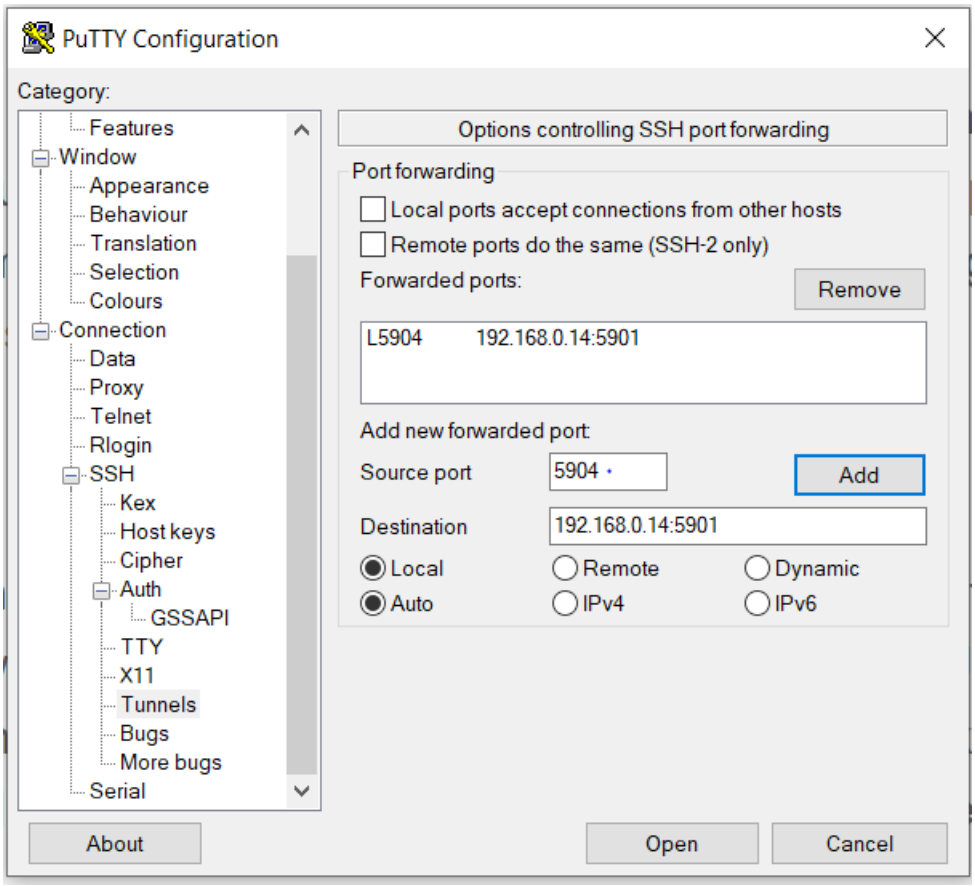
Download PuTTY

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

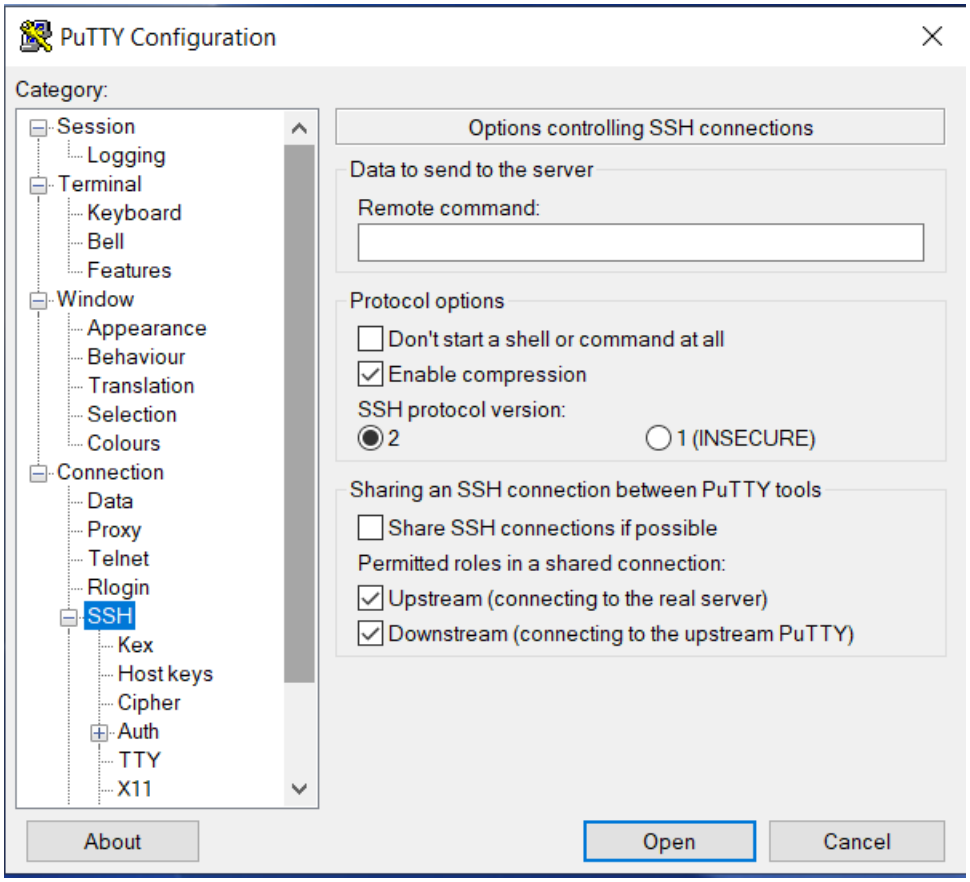
You can download PuTTY here.

B. Once PuTTY is downloaded, configure the SSH connection before adding the tunnel. For more information see Connecting with PuTTy.

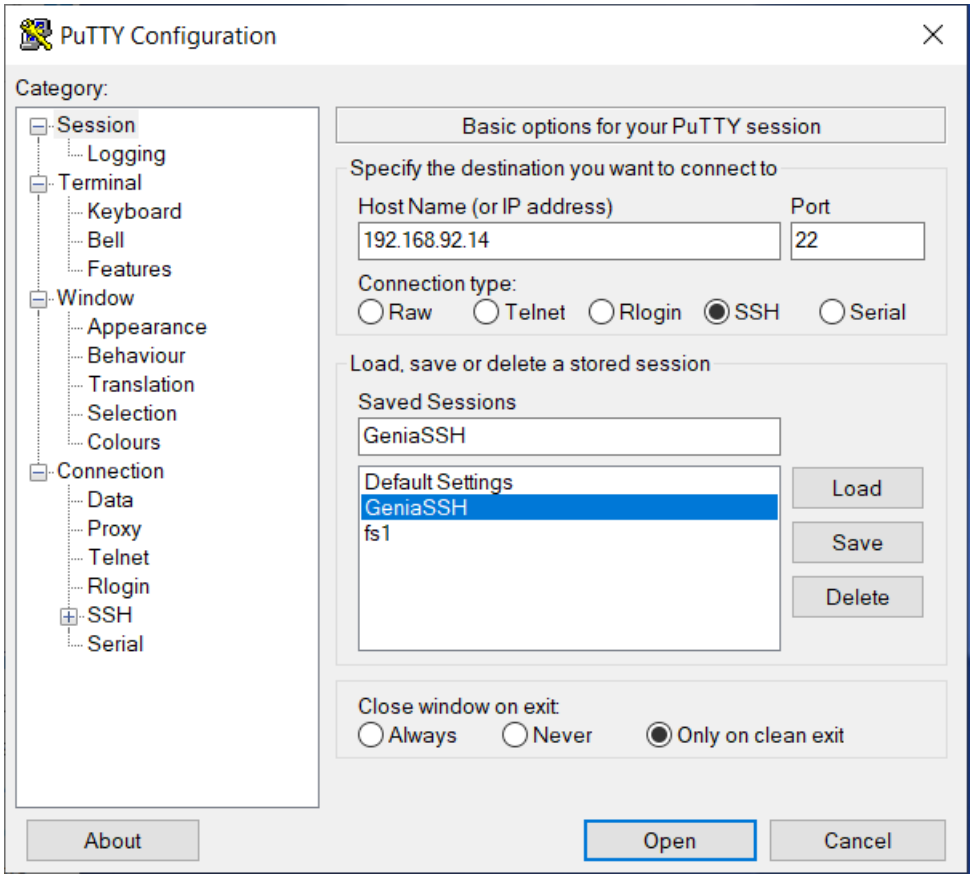
C. Once your session is setup, select your session that was just saved from the last cookbook, then on the left-hand panel, select *Connection -> SSH -> Tunnels*.



D. After setting up the tunnel, select *SSH* and *enable compression*. This will ensure that the tunnel uses data compression.



E. Once all the settings desired are configured, select *Session,* highlight the session again in *Saved Sessions* and hit *Save* for the new session settings. This will make sure that the next time logged in will include all the settings here.



F. Now, the session is saved and can be opened by clicking *Open*