Your own private SSH, VPN and VNC server

A complete solution for taking control of your public access security
An easy path to greater security through
OpenSSH, OpenVPN and VNC on a single-honed Fedora 21 Workstation
on your home network

We all know that public wifi access is a potential playground for would-be hackers with nefarious desires. You can easily take a ***major*** step in protecting yourself by installing an OpenSSH and an OpenVPN server on your home network; through which you encrypt and tunnel all of your public access traffic. By doing so, you will only be providing the coffee house hacker with nothing more than an encrypted stream of data. An encrypted stream that provides absolutely nothing useful to assist him/her in their nefarious deeds.

While there are publicly available VPN and SSH servers available – some free and some not – on the Internet, anyone who has tried to use them has discovered that they are not as reliable as they had hoped them to be: Difficulty in connecting, and very poor performance are common. Many people feel that the servers should not maintain logs; something that is difficult to find without paying a monthly or annual fee – which, if you think about it, takes away your anonymity because now they have a record of their sales transaction.

The best possible solution for this situation is to set up a private SSH and VPN server on your home network and use them when you are out on the road or overseas: You won't have logs to worry about, it is always available and totally ***exclusive*** to you; which means that your performance should be outstanding! And, all of your traffic transverses an encrypted channel which makes it virtually immune to hacking and prying eyes.

If you are more accustomed to using a GUI for administration of your computers, no worries! This article will show you how to set up a VNC server (remote desktop) that you can use to open that server in a window with full GUI access – from anywhere in the world – *and do it via an SSH encrypted tunnel or using your brand new VPN Server!*

You don't need a lot of money or heavy duty equipment to make it work either! Any older computer with at least a Pentium processor, a single connection, via Ethernet ***or*** WiFi, to your home network, a 60 GB (or larger) hard disk, and 1 GB (multi-user.target [runlevel 3]) or 2 GB (graphical.target [runlevel 5]) RAM will suffice. You can probably pick one up for free from a friend that is looking to dump the old “boat anchor” somewhere. All the software required for this project is free, so this minor investment into your private access security is well worth the time and effort to implement.

Project Requirements...

1. Boat-anchor computer (as outlined above)
2. Fedora 21 workstation iso file
3. A blank DVD or a 2GB USB drive
4. About 4 hours of time
5. A ***basic*** understanding of using a terminal; both as root and as a regular user

Hint: use **su** and enter your password to get to root. Use **exit** to get back to the regular user; which is when you will use **sudo** to perform certain tasks. You will never use **sudo** as root.

Caveats and disclaimers...

While this was written during installation and testing on Fedora 21 workstation, the principles involved are the same regardless of distro. If you are not using Fedora 21, then you should still be able to figure out the details with some google searches for your distro. Since there may be documentation errors here that I am not aware of, use these procedures at your own risk! Double check everything, type slowly, deliberately and double check again before pressing Enter!! Do not just copy and paste the commands from this document into a root terminal!! You will most certainly bring your machine down! Note that you will already have installed Fedora 21 Workstation on a computer – you can use whatever partitioning you desire for this project. The main goal is to just get it installed and working on the home network. ***Also, only the Linux client is explained in this article.***

Please note that these procedures are for a freshly installed Fedora 21 Workstation. If you go through these procedures on a box that has had its firewall, iptables, etc. modified from the default values, then you ***may*** run into issues and have to develop workarounds for your particular circumstances ...

One more thing. This work is a conglomeration, with very little of the material being original. I had to dig through countless sites and read until my eyes bled to put together the solution as outlined here; taking bits and pieces from here and there; working my way through all the errors, developing work-arounds, etc.. I apologize to the authors of the information that I have included "wholesale" for not citing each and every procedure/note/comment, etc. that is written here. That would be near impossible. If you see something here that you yourself have written, I know that you understand where I am coming from on this issue.

Food for Thought: Numbering private subnets

Setting up a VPN often entails linking together private subnets from different locations.

For example, suppose you use the popular 192.168.0.0/24 subnet as your private LAN subnet. Now you are trying to connect to the VPN from an internet cafe which is using the same subnet for its WiFi LAN. You will have a routing conflict because your machine won't know if 192.168.0.1 refers to the local WiFi gateway or to the same address on the VPN.

The best solution is to avoid using 10.0.0.0/24 or 192.168.0.0/24 as private LAN network addresses. Instead, use something that has a lower probability of being used in a WiFi cafe, airport, or hotel where you might expect to connect from remotely. ***The best candidates are subnets in the middle of the vast 10.0.0.0/8 netblock (for example 10.66.77.0/24)***.

With this in mind, it is a good idea to start this whole project by changing your home router to provide DHCP addresses in the "middle" of the 10.0.0.0/8 netblock. As an example, I used my birth year to set up my local network addresses; so mine is set as: 10.19.58.0/24

Remember that in all instances within this document you have to substitute user with a valid user on your VPN Server box! The same goes for servername.

Warning: For some unknown reason, if you copy and paste a command line from this document the **–** will not appear when pasted! Therefore, double check your command line and replace the **–** where necessary.

Set up DDNS on the Internet...

Go to http://www.noip.com/remote-access and signup for the free DDNS account.
Download the update client, configure and install it after setting up your router (see below) to allow the ports used by the updater. http://www.noip.com/support/knowledgebase/ should answer all of your questions.

Set up an SSH Server on the machine that will become your VPN Server

Before we get started with anything else, we need to establish our firewall's Default Zone

Set a Default Zone

```
[root@fedora21test ~]# firewall-cmd --list-all-zones
[root@fedora21test ~]# firewall-cmd --get-default-zone
[root@fedora21test ~]# firewall-cmd --set-default-zone=whicheverzoneyoudesire
```

Further info is available at: https://fedoraproject.org/wiki/FirewallD#Using_firewall-cmd

```
[root@fedora21test ~]# yum -y install openssh
```

OpenSSH has a plethora of options, which will not be covered here. All we need is a basic setup for our project. For those who would like to add additional security by disabling root login and implementing certificates for login authentication (*all **highly** recommended*), all the information needed is here: https://wiki.archlinux.org/index.php/SSH_keys#Disabling_password_logins

For now, one thing you ***should*** do is change the listening port for the SSH server since it will be exposed to the internet. You do that by changing the port designation on line 16 of the /etc/ssh/sshd_config file. Of course, you will have to add that port to the server's firewall as well as adjust the entry on the Virtual Servers page of your router (see below). You can add your custom port to the firewall using:

```
[root@fedora21test]# firewall-cmd --permanent --add-port=yoursshportdesignator/tcp
```

Now add the ssh service to the firewall, enable and start the ssh server deamon...

```
[root@fedora21test ~]# firewall-cmd --permanent --add-service ssh
[root@fedora21test ~]# firewall-cmd --reload
[root@fedora21test ~]# systemctl -f enable sshd.service
[root@fedora21test ~]# systemctl daemon-reload
[root@fedora21test ~]# systemctl start sshd.service
```

Tip: install ***htop*** as well to see exactly how your server is doing in reference to cpu load, memory usage and services running, etc. from the terminal used to connect to your server using SSH.

```
[root@fedora21test ~]# yum -y install htop
```

Once installed you will be able to log onto the server with ssh, type ***htop*** at the user prompt, and get a complete, real-time, health report of your server.

To connect to your server locally
[user@fedora21test ~]# ssh user@x.x.x.x ← local ip address of ssh server; i.e. 10.19.58.14

To connect when away from home
[user@fedora21test ~]# ssh user@yourdomain.ddns.net ← whatever your set up is at no-ip.com

You should now be able to administer this server from anywhere on the planet using your newly set up DDNS and SSH! You could just stop right here and configure your applications individually to use the SSH Tunnel, but, the VPN solution is way more secure and actually very easy to set up: You just have to make sure that you make the appropriate changes before you begin the OpenVPN Server installation.

Tip: Add your server name and ip address to the clients' /etc/hosts so that you can just do something like:

[user@fedora21test ~]# ssh user@servername

to make the connection.

Tip: There are a series of scripts at the end of this document that will allow you to connect in various ways after everything is set up and running. Start by creating a ***bin*** directory in your ***home*** directory and copy the contents for each script into the files indicated and **chmod 700** on all scripts. Then you can create a link to each of them and place those links in a folder on your desktop. All of this is outlined in detail later in this article.

Configure your router

Find out where you add Virtual Servers, DHCP Reservations and Port Triggers on your router and add the following...

Virtual Servers – Default settings are used for the servers in this listing

If you decide to change your ports for SSH and VNC then enter those ports instead of the defaults listed here...

Description	Inbound Port	Type	Private IP Address	Local Port
DUC1	8245-8245	Both	10.19.58.14	8245-8245
DUC2	943-943	TCP	10.19.58.14	943-943
VPN1	1194-1194	UDP	10.19.58.14	1194-1194
VPN2	443-443	TCP	10.19.58.14	443-443
SSHServer	22-22	TCP	10.19.58.14	22-22
VNCServer	5910	TCP	10.19.58.14	5910
VNCServer	6010	TCP	10.19.58.14	6010

Note: The 6010 entry is not necessary if you will not be using ip6 to access your vnc server.
The Private IP address is the address that you will reserve, under the DHCP options, for your VPN server box. Unless you are using my ip scheme, it is not going to be 10.19.58.14 for those entries.

Port Triggers – Default settings are used for the servers in this listing

If you decide to change your port for the VNCServer then enter that port instead of the defaults listed here...

Description	Outbound Port	Type	Inbound Port
DUC Out	8245-8245	Both	8245-8245
VNC	5910	TCP	5910
VNC	6010	TCP	6010

Note: The 6010 entry is not necessary if you will not be using ip6 to access your vnc server.
A Port Trigger is needed because these services initiate a connection from inside the network instead of just responding to incoming requests. The VNC entries are required *ONLY* if you plan on doing GUI installs on the server at some later time. See the *Fedora Installation Guide* for details on VNC based installs.

Reserved IP Client List

Name	IP Address	Mac Address	Status
[Server Name]	10.19.58.14	00:03:C0:10:BE:40	Online

Enable IP masquerading

[root@fedora21test ~]# firewall-cmd --permanent --add-masquerade

Then reload the firewall

[root@fedora21test ~]# firewall-cmd --reload

Double check your work...

[root@fedora21test ~]# firewall-cmd --query-masquerade && echo "enabled" || echo "Not enabled"

You should get back a response of:
yes
enabled

Enable IP Forwarding

Next, edit or create /etc/sysctl.d/99-sysctl.conf to permanently enable IPv4 packet forwarding (*takes effect at the next boot*):

[root@fedora21test ~]# vim /etc/sysctl.d/99-sysctl.conf

Enable packet forwarding by putting this line into that file

[root@fedora21test ~]# net.ipv4.ip_forward=1

Double check your work...

[root@fedora21test ~]# sysctl net.ipv4.ip_forward

You should get back a response of:
net.ipv4.ip_forward = 1

Adjust your iptables appropriately

First get your network-interface-id for the next set of entries

[root@fedora21test ~]# ifconfig

Now make these entries:

[root@fedora21test ~]# iptables -A INPUT -i tun+ -j ACCEPT
[root@fedora21test ~]# iptables -A FORWARD -i tun+ -j ACCEPT
[root@fedora21test ~]# iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o network-interface-id -j MASQUERADE

Adjust SELinux Policy

[root@fedora21test ~]# getenforce

[root@fedora21test ~]# vim /etc/selinux/config

change SELINUX=enforcing to SELINUX=*disabled* or *permissive* (see note below)

[root@fedora21test ~]# reboot

Note: ***Disabling*** selinux is required for the VNC server installation. If you do not intend on installing the VNC server, then you can set the above variable declaration to *permissive*.

Install OpenVPN

Install OpenVPN on the server

[root@fedora21test ~]# yum install openvpn -y

OpenVPN ships with a sample server configuration, so we will copy it to where we need it:

[root@fedora21test ~]# cp /usr/share/doc/openvpn/sample/sample-config-files/server.conf /etc/openvpn

Generate Keys and Certificates

NOTE: Build your keys on the machine with the highest Random Entropy. You do not have to do this on the server. To obtain your random entropy figure, use: *cat /proc/sys/kernel/random/entropy_avail (the higher the figure the better. Do NOT use a machine that has less than 200!) See this article for more information on random entropy and its importance in encryption: https://major.io/2007/07/01/check-available-entropy-in-linux/ The comments section has solutions for generating higher random entropy.*

Also, note that the server and client clocks need to be roughly in sync or certificates might not work properly. If not already set up, you should use ***ntp*** *on both your server and clients.*

Generate the master Certificate Authority (CA) certificate & key

Install easy-rsa

[root@fedora21test ~]# yum install easy-rsa

Copy the easy-rsa directory to /etc/openvpn/

[root@fedora21test ~]# cp -r /usr/share/easy-rsa /etc/openvpn
[root@fedora21test ~]# cd /etc/openvpn/easy-rsa
[root@fedora21test ~]# init-config

Now edit the **vars** file and set the KEY_COUNTRY, KEY_PROVINCE, KEY_CITY, KEY_ORG, and KEY_EMAIL parameters. Don't leave any of these parameters blank.

Next, initialize PKI:

[root@fedora21test ~]# . ./vars
[root@fedora21test ~]# ./clean-all

```
[root@fedora21test ~]#  ./build-ca
```

The final command (**build-ca**) will build the certificate authority (CA) certificate and key by invoking the interactive **openssl** command:

```
ai:easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
.....+++++++
.....+++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [KG]:
State or Province Name (full name) [NA]:
Locality Name (eg, city) [BISHKEK]:
Organization Name (eg, company) [OpenVPN-TEST]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []: Servername-CA
Email Address [me@myhost.mydomain]:
```

Note that in the above sequence, most queried parameters were defaulted to the values set in the **vars** or **vars.bat** files. The only parameter which must be explicitly entered is the **Common Name**. In the example above, I used Servername-CA which you should change to reflect your own CA name.

Generate certificate & key for server

```
[root@fedora21test ~]#  ./build-key-server server
```

As in the previous step, most parameters can be defaulted. When the **Common Name** is queried, enter "server". Two other queries require positive responses:

"Sign the certificate? [y/n]"
"1 out of 1 certificate requests certified, commit? [y/n]".

Generate certificates & keys for clients

```
[root@fedora21test ~]# ./build-key client1
[root@fedora21test ~]# ./build-key client2
[root@fedora21test ~]# ./build-key client3 (etc.)
```

Remember that for each client, make sure to type the appropriate **Common Name** when prompted, i.e. "client1", "client2", or "client3". Always use a unique common name for each client.

Generate Diffie Hellman

[root@fedora21test ~]# ./build-dh

Generate the ta.key file

[root@fedora21test ~]# openvpn --genkey --secret ta.key

Key Files Summary

All of your newly-generated keys and certificates in the /etc/openvpn/easy-rsa/keys sub-directory.

Here is an explanation of the relevant files:

Filename	Needed By	Purpose	Secret
ca.crt	server + all clients	Root CA certificate	NO
ca.key	key signing machine only	Root CA key	YES
dh{n}.pem	server only	Diffie Hellman parameters	NO
server.crt	server only	Server Certificate	NO
server.key	server only	Server Key	YES
ta.key	server + all clients	tls-auth	YES
client1.crt	client1 only	Client1 Certificate	NO
client1.key	client1 only	Client1 Key	YES
client2.crt	client2 only	Client2 Certificate	NO
client2.key	client2 only	Client2 Key	YES
client3.crt	client3 only	Client3 Certificate	NO
client3.key	client3 only	Client3 Key	YES

Distribute keys

[root@fedora21test ~]# mkdir /etc/openvpn/keys

The final step is to copy the appropriate files to the /etc/openvpn/keys directory of machines that need them. Take extra care to copy secret files over a secure channel (or USB) to the other computers.

Although you can place the keys anywhere, for the sample config files to work, all keys are placed in the same sub-directory on all machines: /etc/openvpn/keys

Once the keys are in place you need to update your selinux policy on each of the clients and the server. This is not required if you set your policy to disabled.

```
[root@fedora21test ~]# restorecon -Rv /etc/openvpn
```

Edit the server configuration file
(A completed, working sample file is included in this document)

The default OpenVPN server configuration will create a **tun0** network interface (for routing), will listen for client connections on **UDP port 1194** (OpenVPN's default), authenticate client access, and distribute virtual addresses to connecting clients from the **10.8.0.0/24** subnet.

```
[root@fedora21test ~]# vim /etc/openvpn/server.conf
```

Edit the **ca**, **cert**, **key**, **tls-auth** and **dh** parameters to point to the ca, key, cert, dh and ta files you generated above. (i.e. /etc/openvpn/keys) and uncomment the following items:

```
server
proto udp
dev tun
topology subnet
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
cipher ← choose which method you desire and leave the others commented.
user nobody
group nobody
```

Install OpenVPN on the client

```
[root@fedora21Client ~]#  yum install openvpn -y
```

Edit the client configuration file
(A completed sample file is included in this document)

The sample client configuration file (**client.conf**) mirrors the default directives set in the sample server configuration file. Since OpenVPN can be either a server or client, configuration files for both are included when installed.

```
[root@fedora21Client ~]#  cp /usr/share/doc/openvpn/sample/sample-config-files/client.conf /etc/openvpn
[root@fedora21Client ~]# vim /etc/openvpn/client.conf
```

Edit the **ca**, **cert**, **key**, and **tls-auth** parameters to point to the ca, key, cert, and ta files you generated above. (i.e. /etc/openvpn/keys) and check the following items:

1. The **remote** directive points to the hostname/IP address and port number of your OpenVPN server.
Follow the instructions above for installation, configuration and setup of DDNS.
You will use your DDNS name on the **remote** line...

2. That the **dev** (tun), **proto** (udp), **cipher, comp-lzo** and **fragment** directives are consistent with the server.conf file.

3. That the ***remote-cert-tls server*** option is uncommented.

Final Steps for server

[root@fedora21test ~]# ln -s /lib/systemd/system/openvpn\@.service /etc/systemd/system/multi-user.target.wants/openvpn\@server.service
[root@fedora21test ~]# firewall-cmd --permanent --add-service openvpn
[root@fedora21test ~]# systemctl stop openvpn@server.service
[root@fedora21test ~]# systemctl -f enable openvpn@server.service
[root@fedora21test ~]# systemctl start openvpn@server.service

Final Steps for clients

[root@fedora21Client ~]# firewall-cmd --permanent --zone=FedoraWorkstation --add-service openvpn
[root@fedora21Client ~]# firewall-cmd --permanent --zone=FedoraWorkstation --add-service ssh

Either run the VPN Client via command line in root terminal (Ctrl-c will exit) or set up your VPN connection using Network Manager (covered below). For now, we will just use a terminal...

To open a VPN connection using the command line:

[root@fedora21Client ~]# openvpn /etc/openvpn/client.conf

Test the connection by pinging your side of the tunnel, the VPN server's side of the tunnel (usually at 10.8.0.1) and the internal ip address of the VPN server (the one it uses to communicate on its Ethernet or WiFi connection to the router) and finally the default gateway of the VPN server itself.

[root@fedora21Client ~]# ifconfig ← to get ip address of tun0
[root@fedora21Client ~]# ping 10.8.0.2 (tun0 ip address)
[root@fedora21Client ~]# ping 10.8.0.1 (vpn server tun0 address)
[root@fedora21Client ~]# ping 10.19.58.14 (vpn server address)
[root@fedora21Client ~]# ping 10.19.58.1 (vpn server default gateway)

Ensure that all traffic is flowing through the tunnel:

[root@fedora21Client ~]# ip route get [the ip address you used in testing the connection] one ip at a time.

Note: The default gateway of the router may come back with a good ping, but *ip route get* may show a route other than the tunnel. In that case, the easiest way to test is to connect your laptop wifi to your cell phone hotspot (or connect to some other external wifi network), reconnect to the VPN server, and run the ***ping*** & ***ip route get*** commands again.

When successful, whatismyip.com should show the ip address of your home routers external ip.
When you disconnect it should show your cell phones ip address.

Once you have verified that you can ping successfully through the vpn connection, you can run **traceroute** and **ip route get** to verify that ***all*** traffic is flowing through the tunnel.
Compare name resolution for both 1. While vpn is up and 2. When VPN is down.

While VPN is UP

[root@fedora21Client ~]# traceroute www.google.com
Which should return a lot of information, but the most important part is in the first two lines:

1 10.8.0.1 (10.8.0.1) 58.430 ms 68.225 ms 77.327 ms
2 10.19.58.1 (10.19.58.1) 77.510 ms 68.233 ms 68.235 ms

Notice that the first route is to the vpn server and the second is to the default gateway on the server.

```
[root@fedora21Client ~]#  ip route get 8.8.8.8
8.8.8.8 via 10.8.0.1 dev tun0  src 10.8.0.2
```

Notice that it shows that the route was obtained through your tun0 interface and that it got that route via the vpn server's tun0 interface address.

When VPN is DOWN

```
[root@fedora21Client ~]# traceroute www.microsoft.com
traceroute to www.microsoft.com (23.66.56.154), 30 hops max, 60 byte packets
 1  192.168.43.1 (192.168.43.1)  6.727 ms  10.024 ms  10.031 ms
 2  33.sub-66-174-43.myvzw.com (66.174.43.33)  43.569 ms  43.579 ms  53.647 ms
```

Useful OpenVPN References:

https://www.digitalocean.com/community/tutorials/how-to-setup-and-configure-an-openvpn-server-on-centos-6

https://openvpn.net/index.php/open-source/documentation/howto.html#numbering

https://wiki.archlinux.org/index.php/OpenVPN

Contents of /lib/systemd/system/openvpn@.server

```
[Unit]
Description=OpenVPN on %I
After=network.target

[Service]
PrivateTmp=true
Type=forking
PIDFile=/var/run/openvpn/%i.pid
ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i
.pid --cd /etc/openvpn/ --config %i.conf

[Install]
WantedBy=multi-user.target
```

You can use this text for your ***server.conf*** if you have been following the procedures used in this document, or modify it if your keys are in different locations or you have made other adjustments along the way.

Example of working, simplified /etc/openvpn/server.conf – All comments removed...

```
server
port 1194
proto udp
dev tun
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/server.crt
key /etc/openvpn/keys/server.key
dh /etc/openvpn/keys/dh2048.pem
topology subnet
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
keepalive 10 120
tls-auth /etc/openvpn/keys/ta.key 0
cipher BF-CBC
comp-lzo
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
verb 3
```

If you choose not to keep status logs, then comment out the status openvpn-status.log and the verb 3 lines.

You can use this text for your ***client.conf*** if you have been following the procedures used in this document, or modify it if your keys are in different locations or you have made other adjustments along the way.
Just make sure to insert your own DDNS value where the text is red in the following:

Example of working, simplified /etc/openvpn/client.conf – All comments removed...

```
client
dev tun
proto udp
remote yourdomain.ddns.net-1 1194
resolv-retry infinite
nobind
user nobody
group nobody
persist-key
persist-tun
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/client1.crt
key /etc/openvpn/keys/client1.key
remote-cert-tls server
tls-auth /etc/openvpn/keys/ta.key 1
cipher BF-CBC
comp-lzo
verb 3
```

If you choose not to keep status logs, then comment out the verb 3 line.

Tip: Once you have created and saved your client.conf file you can import it into Network Manager! Just bring up Network Connections, select add, under VPN select ***Import a saved OpenVPN configuration***, point to your /etc/openvpn/client.conf file and hit create!

Double check all of the *Authentication* information on the *VPN* tab (you do not need to enter anything in the password box), then select *Advanced* and verify the information on the *TLS Authentication* tab. Save and exit.

Now, you will be able toggle your VPN in Network Manager!

VNC Server and Client Setup – With instructions for using VNC over SSH

This may seem pretty obvious, but VNC Server will not load if you boot your computer into multi-user.target (runlevel 3).
To change back and forth you can run one of the following and then reboot.

[root@fedora21test ~]# systemctl set-default multi-user.target
[root@fedora21test ~]# systemctl set-default graphical.target

Be aware that you **may** have to re-do the server configuration from scratch if you switch to multi-user.target and then want to go back to graphical.target. Shortened procedures for doing this are near the end of this document.

NOTE! Pay very close attention to *which* user is issuing the command!
This changing back and forth is necessary for the appropriate files to be created in the correct locations!
Watch what you are doing very closely here because it is very easy to get fouled up.

Step 1 :

Install the Tiger VNC server & client packages

[root@fedora21test ~]# yum -y install tigervnc-server (on the server)
[root@fedora21test ~]# yum -y install tigervnc (on the client)

Step 2 :

Copy the VNC server configuration file to where it needs to be for editing:

[root@fedora21test ~]# cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@:10.service

By default, VNC server uses port 5900. The vncserver@:10.service file, shows a port-offset of 10. This means that the service will be listening on the base server listening port (5900) + 10 = 5910. So, X11 will use port 5910 for routing the display to the client. This means we are running the service on a sub-port of the default port 5900. When using this port-offset we can connect to the VNC server by specifying the IP address:sub-port format.
Eg: 10.30.0.78:10*

* **Tip:** For added security, you can change the listening port to whatever you desire on line 199 of the /usr/bin/vncserver file

[root@fedora21test ~]# vim /usr/bin/vncserver

199 $vncPort = 4400 + $displayNumber;

In this example the listening port has been changed to 4400. Since the @:10.service file shows a 10 offset, you would use 4410 instead of 5910 in the upcoming firewall commands. You would also need to ensure that you are using 4410 on the Virtual Server page of your router for the VNC Server.

Step 3:

Edit the copied file and make changes as mentioned below. Changes are indicated in *RED* and ***BOLD*** below...

[root@fedora21test ~]# vim /etc/systemd/system/vncserver@\:10.service

For simplicity sake, I have a user named “user”, so in my example this user is what the server will authenticate me against when attempting to connect. Thus, my file would look the one below:

Please choose your user and change accordingly.

```
[Unit]
Description=Remote desktop service (VNC)
After=syslog.target network.target
[Service]
Type=forked    #change this to simple if you are having issues with loading
# Clean any existing files in /tmp/.X11-unix !!
ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'
ExecStart=/sbin/runuser -l user -c "/usr/bin/vncserver %i"
PIDFile=/home/user/.vnc/%H%i.pid
ExecStop=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'
[Install]
WantedBy=multi-user.target
```

Step 4:

Now add the service and ports to the firewall for VNC.

```
[root@fedora21test ~]# firewall-cmd --permanent --add-service vnc-server
[root@fedora21test ~]# firewall-cmd --permanent --add-port=5910/tcp
[root@fedora21test ~]# firewall-cmd --permanent --add-port=6010/tcp
```

Remember to change the port here to the one you entered in /usr/bin/vncserver.
Also, the 6010 entry is not necessary if you are not going to use ip6 for VNC.

NOTE: If you haven't already done do, don't forget to add these ports to the ***Virtual Server*** section on your router (as shown under ***Configure your Router*** above). If you intend on doing installs/upgrades via VNC on this same box (i.e. Fedora Server using the GUI interface – which is recommended) then you will also have to add a ***Port Trigger*** for these two addresses; since in that case the server will be initiating a connection with a listening client... See the ***Fedora Installation guide*** for details. *Once again, the 6010 entry will not necessary on the Port Trigger page if you are not going to use ip6 for VNC.*

Now, reload the firewall with:

```
[root@fedora21test ~]# firewall-cmd --reload
```

Hints: You can always double-check which ports are open, and which services are allowed, on the firewall with:

```
[root@fedora21test ~]# firewall-cmd --list-ports
[root@fedora21test ~]# firewall-cmd --get-services
```

For a complete listing of all firewall-cmd's refer to this site:
https://fedoraproject.org/wiki/FirewallD#Using_firewall-cmd

Step 5:

Next, setup a password for the VNC user.

Using a standard (non-root) terminal, do as indicated below:

```
[user@fedora21test ~]$ vncpasswd
Password :
Verify :
```

After this process has finished, a new directory (.vnc) will be created under the home directory of the user with a ***passwd*** file in it.

Check to make sure it was created...

```
[user@fedora21test ~]# ls -l /home/user/.vnc/
-rw-------. 1 user user 8 Feb 20 17:55 passwd
```

Step 6 :

Now reload the systemctl daemon and start the VNC service.

```
[user@fedora21test ~]# sudo systemctl daemon-reload
[user@fedora21test ~]# sudo systemctl -f enable vncserver@:10.service
[user@fedora21test ~]# sudo systemctl start vncserver@:10.service.
```

After the server service successfully starts, you can verify which ports the VNCServer is listening to:

```
[root@fedora21test ~]# lsof -i -P | grep -i "listen" | grep Xvn
Xvnc      8433    vpn    7u  IPv4 110262      0t0  TCP *:5910 (LISTEN)
```

Running the ***systemctl start*** above will create an ***xstartup*** script under the /home/user/.vnc/ directory of the specific user account.

Check to make sure that it was created...

```
[user@fedora21test ~]# ls -l /home/user/.vnc/
-rw-------. 1 user user 8 Feb 20 17:55 passwd
-rwxr-xr-x. 1 user user 355 Feb 20 17:11 xstartup
```

Step 7 :

IF you need to set the resolution for the VNC desktop, you can edit /etc/sysconfig/vncservers

```
[root@fedora21test ~]# vim /etc/sysconfig/vncservers
```

After editing the configuration file, you will need restart the VNC service.

```
[user@fedora21test ~]# sudo systemctl daemon-reload
[user@fedora21test ~]# sudo systemctl stop vncserver@:10.service.
[user@fedora21test ~]# sudo systemctl start vncserver@:10.service.
```

Step 8

One of the drawbacks of VNC is that its connections and traffic are not encrypted.
The easiest (and most secure) way to use VNC is to run it after you have established a VPN connection.

If you desire to run all of your traffic through an SSH tunnel instead, you can do so by following the procedures outlined below. *Numbers in red need to be changed if you chose to change your ports earlier*

To run your VNC client through a secure ssh tunnel, do the following:

For local network connections...

```
[user@fedora21Client ~]# ssh user@x.x.x.x -L 6999:localhost:5910
```
← x.x.x.x is the local network's ip address for the VNC Server

For cross-internet connections...

[user@fedora21Client ~]# ssh user@mydynamic.ddns.net -L 6999:localhost:5910 ← mydynamic.ddns.net is what was set up at ***noip.com*** and points to your home routers external ip address.
NOTE: You need to connect to the server via SSH every time before running the VNC client...

The above lines essentially say:

Establish an SSH connection and then take all traffic bound for port **6999** on the ***localhost*** *interface*, and forward it through the SSH connection to port **5910** on the server.

Step 9

Regardless of whether you have chosen to use the internal ip address or the external one (as shown above) to forward your VNC traffic over SSH, you will always use the same line (indicated below) to connect via the VNC client.

Now connect using a VNC client...

On the address line enter:

localhost:6999

You should now get a password prompt box.

Enter your password and you will get a pop-up window showing the screen of the server.

Now your VNC connection is running inside of a secure SSH tunnel! WooHoo! :)

Scripts for starting connections to your VPN, SSH, VNC server.

Obviously, you need to edit these scripts with your own particular user name, server name, no-ip domain name, and ip address: highlighted in red below...

ExternalSecureVNC:

```
#!/bin/bash
# To be used prior to using vnc viewer securely across the internet.
ssh user@yourdomain.ddns.net -L 6999:localhost:5910
```

LocalSecureVNC:

```
#!/bin/bash
# To be used prior to using vnc viewer securely locally.
# for a straight ip to the server
;ssh user@x.x.x.x -L 6999:localhost:5910
# for using your /etc/hosts entries
;ssh user@servername -L 6999:localhost:5910
```

StraightSSH:

```
#!/bin/bash
# for a straight local ip to the server
;ssh user@x.x.x.x
# for using your /etc/hosts entries
;ssh user@servername
# for connecting across the internet
;ssh user@yourdomain.ddns.net
```

VPNConnect:

```
#!/bin/bash
sudo openvpn /etc/openvpn/client.conf
```

Notes:

Don't forget to ***chmod 700 ** in the directory where all the scripts have been placed; to allow them to be executable.

It is recommended that you create /home/user/**bin** and put all of your scripts there. That way they will run from any location. While using the terminal, simply type the name of the script.

As an alternative, you can create links to your scripts and put them either on the desktop (or into a folder on the desktop) for easy access and execution. This can be accomplished by executing the following:

```
[user@fedora21test ~]# ln -s /home/user/bin/nameofscript  /home/user/Desktop/nameofscript
```

What to do if the VNC Server Service fails to start...

If the server fails to start for whatever reason, the following procedures should get it going ...

First check to see if this file still exists:

[root@fedora21test ~]# ls /etc/systemd/system/vncserver@\:10.service

If not, then copy and edit the sample file again and replace <user> with a valid user on your system:

[root@fedora21test ~]# cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@:10.service

[root@fedora21test ~]# vim /etc/systemd/system/vncserver@\:10.service

Then continue with the following commands:

[user@fedora21test ~]# sudo rm -f /home/user/.vnc/*
[user@fedora21test ~]# sudo rm -f /tmp/.X11-unix/*
[user@fedora21test ~]# vncpasswd ← Do ***NOT*** use root or sudo here.
enter the vncserver password twice

Display issues can usually be cleared up by editing the vncserver file and making the following change (I have indicated the line numbers below)**:**

[root@fedora21test ~]# vim /usr/bin/vncserver

Go to this section and uncomment the line &GetXDisplayDefaults();

113 # Uncomment this line if you want default geometry, depth and pixelformat
114 # to match the current X display:
115 &GetXDisplayDefaults();

:wq

[user@fedora21test ~]# sudo systemctl daemon-reload
[user@fedora21test ~]# sudo systemctl -f enable vncserver@:10.service
[user@fedora21test ~]# sudo systemctl start vncserver@:10.service

Now go back and check that there is a new xstartup script and passwd file in /home/user/.vnc
as well as new entries in /tmp/.X11-unix

[user@fedora21test ~]# ls -a /home/user/.vnc
[user@fedora21test ~]# sudo ls -a /tmp/.X11-unix

If everything went according to plan, you should now have the ability to be anywhere on the planet and have a secure communication channel back to your home server via SSH, VPN and VNC! Each one of these services provides you with different and various options to access your server, or pass through it out on to the Internet, securely. Do your homework on each of these services and stay safe!

What to do if the client just goes away after clicking connect...

```
[root@fedora21test ~]# yum erase tigervnc
[root@fedora21test ~]# reboot
[root@fedora21test ~]# yum install tigervnc
```

Double check that your scripts (if you are using them) have the correct port entries:

Bring up tigervnc and if using SSH to provide a secure tunnel for you VNC session, ensure that both of your ssh start scripts for VNC have the correct ports entered.

Remember that you always have to run either the **LocalSecureVNC or ExternalSecureVNC** script before attempting to connect with the vncviewer if you want SSH security.

On the address line of the vncviewer, ensure that you are entering localhost:6999 before selecting connect.

If you want to connect to a VNC server and you have a VPN connection up, ensure that you enter x.x.x.x:portnumber on the address line of the client before selecting connect.

x.x.x.x is the INTERNAL (home network; i.e. 10.19.58.14) ip address of the VNC server: It is the SAME address that you entered on the Virtual Servers page on the router. The portnumber is your chosen port number, if you have changed it from the default, or 5910 if you have not.

The included /etc/openvpn/server.conf has topology subnet as one of the options. If you did not use that config file, then ensure that you have uncommented that entry in the server.conf file you are using. That entry makes it possible to connect through VPN to the VNC server as if it were on the same subnet as you. If you did have to make that change then you will also have to stop and restart the VPN server.

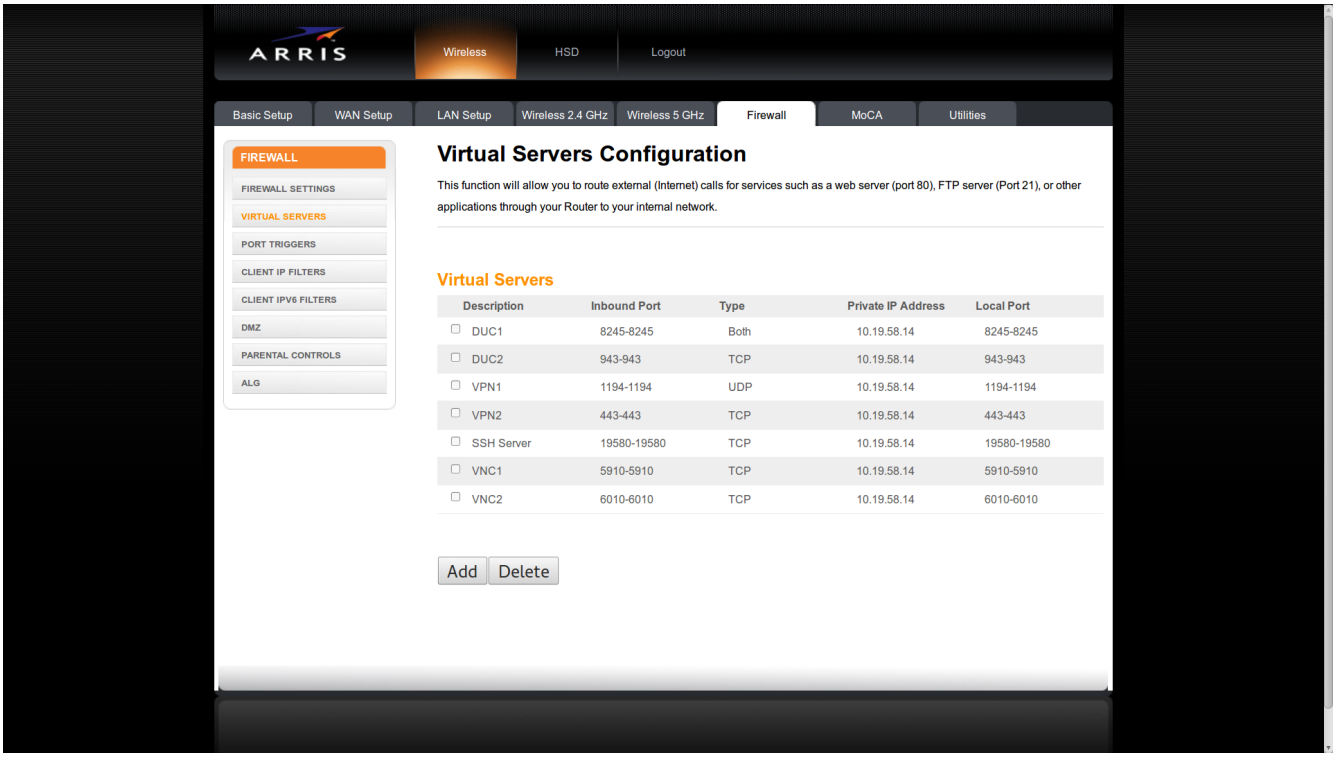
It is much easier to just bring up a VPN connection and then initiate a VNC connection to the server. I have included the SSH procedures as a means of showing an alternative, should the VPN Service be down for any reason.

Additional Tidbits:

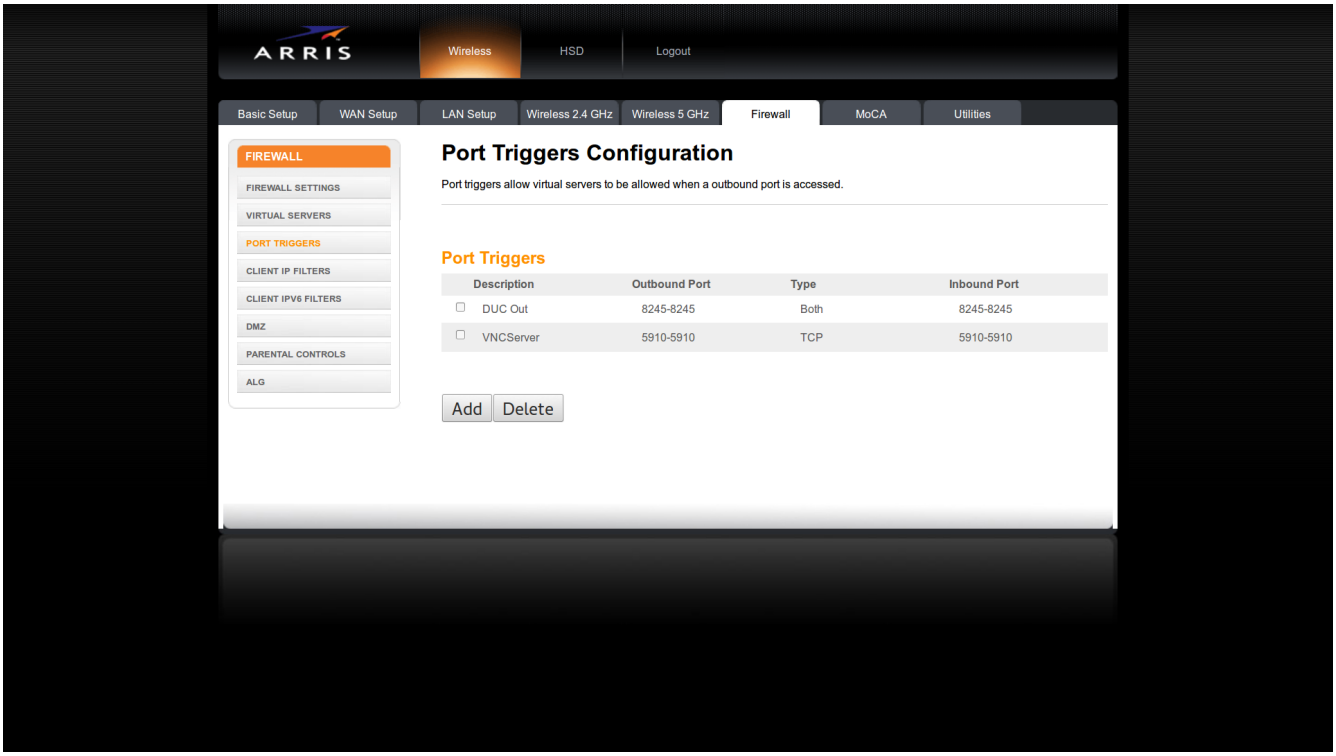
Disk space usage with my own setup:

Filesystem	Size	Used	Avail	Use%	Mounted on
devtmpfs	3.7G	0	3.7G	0%	/dev
tmpfs	3.7G	0	3.7G	0%	/dev/shm
tmpfs	3.7G	856k	3.7G	1%	/run
tmpfs	3.7G	0	3.7G	0%	/sys/fs/cgroup
/dev/mapper/fedora-root	50G	103M	47G	1%	/root
/dev/mapper/fedora-usr	50G	4.3G	43G	10%	/user
/dev/mapper/fedora-home	50G	139M	47G	1%	/home
/dev/mapper/fedora-var	50G	1.5G	46G	4%	/var
/dev/sda2	477M	117M	331M	27%	/boot
/dev/sda1	200M	9.6M	191M	5%	/boot/efi
/dev/mapper/fedora-tmp	20G	45M	19G	1%	/tmp
tmpfs	744M	0	744M	0%	/run/user/1000

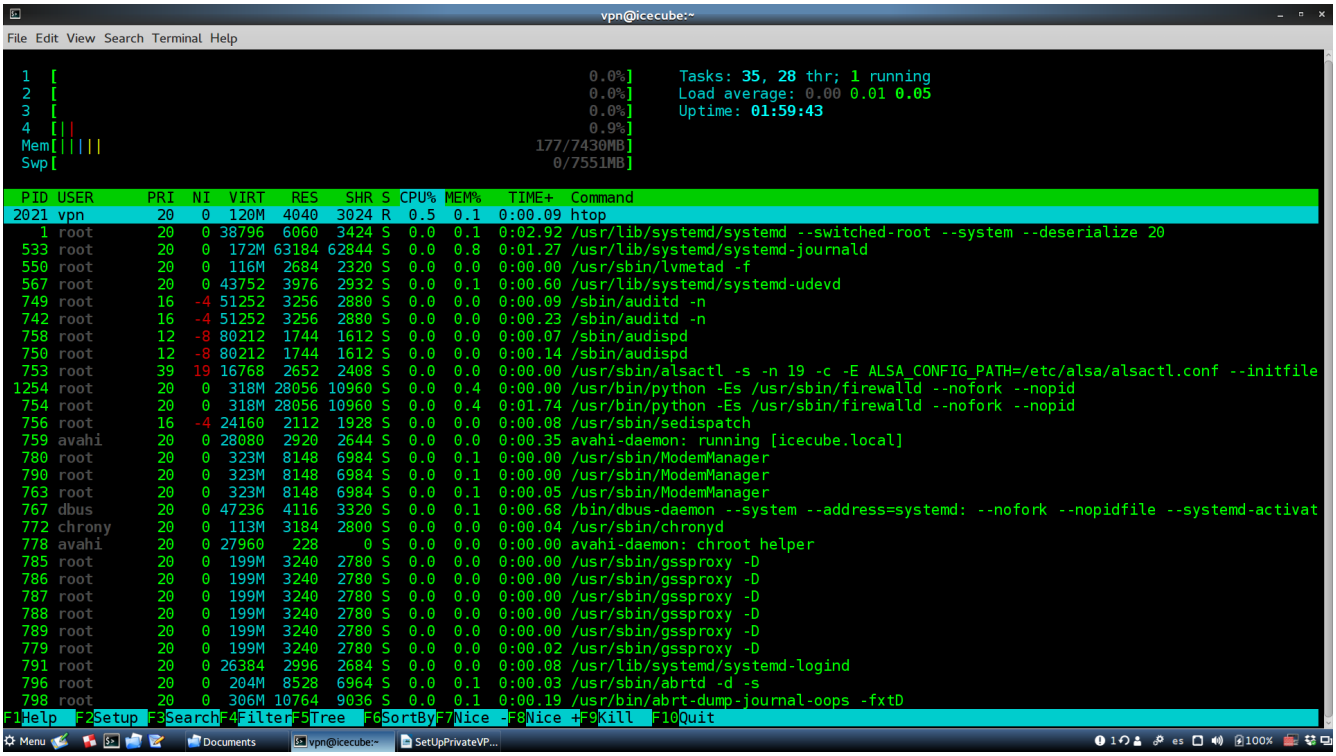
As you can see, my partitioning for this workstation is a bit much for what is actually required...



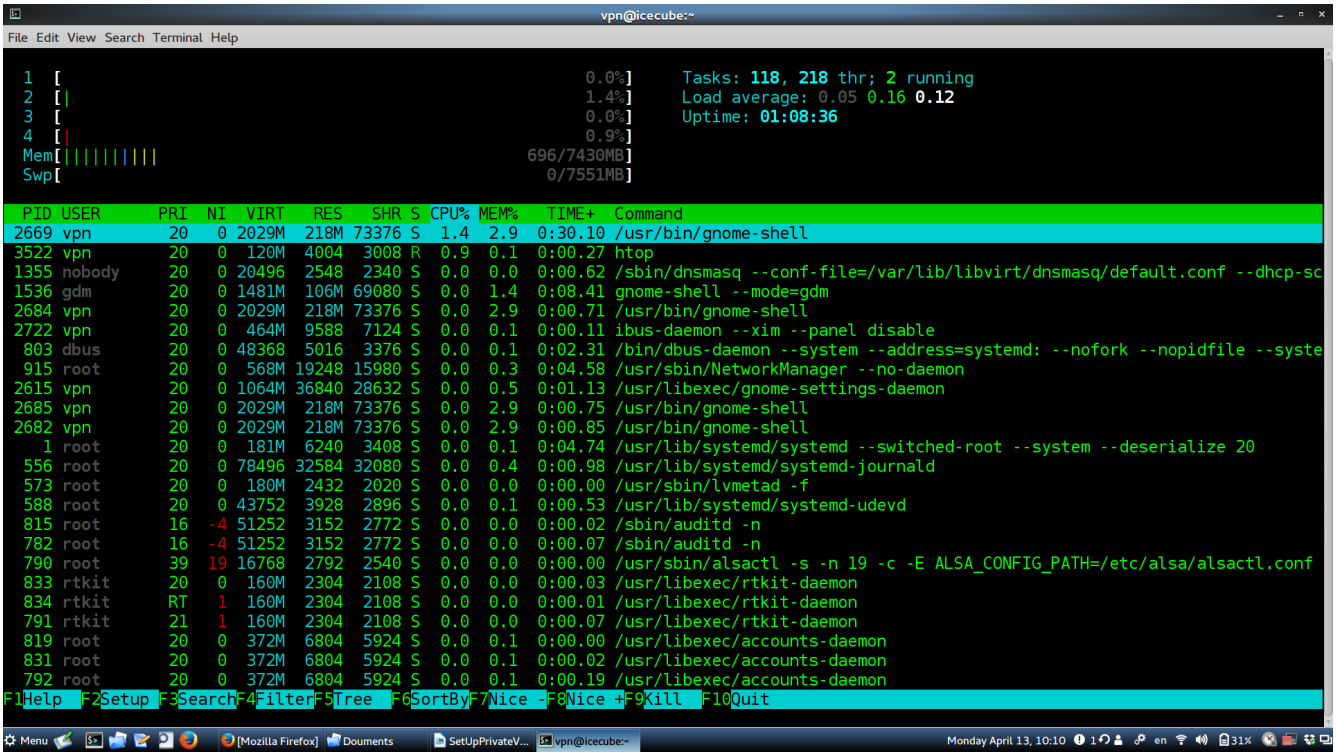
My Virtual Server settings page on the router. Note the changed port number for the SSH server.



The Port Triggers page on my router. Since I will not be using ip6 I have left out the entry for that on this page.



Using htop to view server status via SSH connection over the internet.
Notice that in multi-user.target (runlevel 3) mode, the RAM usage is only 177MB!



But, even in graphical.target mode (runlevel 5), RAM usage is still only 696MB! In either target, performance is ***outstanding*** when connecting externally.

Enjoy!
Warm Regards,
RedBrick