Building a PBS Professional Virtual Test Cluster with Ubuntu LTS

Derrick Hendricks
Technical Specialist – Altair PBS Works™

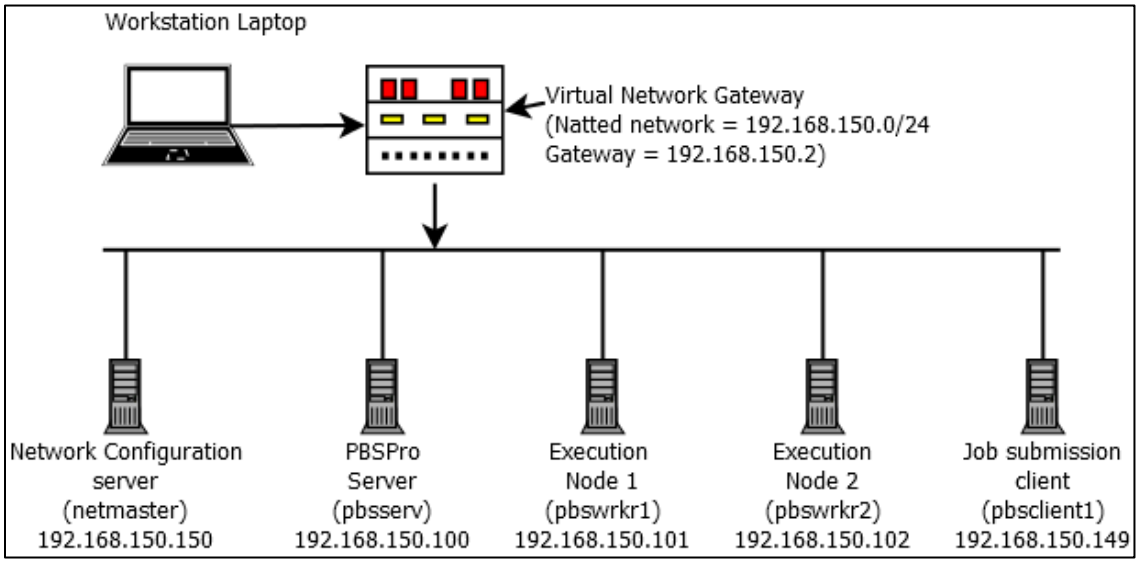
Your PBS Professional Test Platform

Because it isn't always convenient to explore software in a production environment, Altair PBS Professional™ has many features that allow users to simulate huge computing node environments on relatively simple, inexpensive hardware. While you can set up a small physical cluster when you need a test platform, that becomes inconvenient if you want to work on it from another location. What you can do instead is set up a cluster of virtual machines (VMs). VMs are easy to set up.

Step-by-step Instructions

These instructions cover a quick, step-by-step process to get PBS Professional running on a small cluster of five virtual machines with Ubuntu LTS. For this setup we used Ubuntu LTS 18.04. Any VM software that can create a virtual network should be fine. For the purpose of these instructions we used VMware. Separate instructions are available to show you how to use CentOS and SLES for hosting a PBS Professional test cluster.

The first step is to set up your five virtual machines. The diagram below shows how the network will be laid out. Keep in mind that all VM setups are different.



For this setup, we will use five VMs. This is not required; an entire PBS Professional setup can be done on as little as a single VM. There are some advantages to using multiple VMs, though. It's recommended to split off the netmaster from the rest of the PBS Professional machines. That way, the entire PBS Professional installed cluster can be scrapped and rebuilt without needing to rebuild the user directories, DNS, or NFS, so using at least two machines is a good idea. For slightly more extensive testing, it's good to have a client machine also. That makes three machines. To split into at least one actual execution node, you'll need four VMs. In this case, it might be useful to have another execution node for testing a multi-machine setup without having to use vnodes, so add another worker node as a fifth VM.

You'll want a network that is easy to modify and provides the consistency PBS Professional requires. Using a "netmaster" server separates all the normal jobs of network maintenance from the PBS Professional server. It will be responsible for

holding the NFS mounts for the user home directories, running a DNS server so the rest of the machines can resolve the entire cluster, and being a NIS server so the users and passwords are consistent (including UIDs) across the entire network. Technically, this can all be done on the pbsserv machine, but in this configuration you're free to completely wipe out and/or upgrade the PBS Professional server without ever needing to worry about changing the configuration of the rest of the network. Plus, if you want to experiment with other OSes for PBS Professional servers and nodes, you don't have to figure out how to configure the different server daemons in new OSes.

For the purposes of this walk-through, some optional conventions were adopted. We recommend that you read each section in its entirety before following the steps for that section.

- testnet.net was chosen as the domain for the test network. This could be anything, provided it is consistent across all machines in the cluster.
- pbsserv was chosen for the hostname of the server (head node, etc.)
- pbswrkr1 and pbswrkr2 were chosen for the hostnames of the execute servers (compute servers, etc).
- 192.168.150.0/24 was chosen for the network IP range for the NAT'ed network where the VMs reside.
- All five VMs and their network settings are listed in this chart. PBS Professional makes no restrictions on VM names, hostnames, or IP addresses:

Virtual Machines and Network Settings

VM Name	Hostname	IP Address	Description
Netmaster	netmaster.testnet.net	192.168.150.150	Network controlling server
PBSServer	pbsserv.testnet.net	192.168.150.100	PBS server and scheduler
PBSwrkr1	pbswrkr1.testnet.net	192.168.150.101	PBS MoM node 1
PBSwrkr2	pbswrkr2.testnet.net	192.168.150.102	PBS MoM node 2
PBSClient1	pbsclient1.testnet.net	192.168.150.149	PBS client execution machine

- PBS Professional needs all the machines in the cluster to be able to resolve each other. Using static IP addresses makes this easier, but DHCP reservations is also fine.
- PBSTESTNIS was used as the NIS domain. Anything can be used. It does not have to be the same as your DNS domain.
- The pbsdata account UID was designated as 750. This was mostly arbitrary. In fact, the UID can be left to be chosen by the system, but in this case we wanted the NIS server to pick it up also.

The user of this document should always be logged into the VMs as the root user unless specifically stated otherwise.

Preconfigure Daemons (Run on All VMs)

At this point, you have five VMs set up with a basic install of Ubuntu LTS 18.04. You can use the "quick install" method in VMware to make the install go quickly and not have to babysit it. Your VMs can have dynamic IPs and hostnames according to the table above. In the case of the DNS server entries in the installer's network setup, set the netmaster to resolve addresses using your provider's DNS servers. The other four machines should have their DNS entries set to the IP address

of netmaster (192.168.150.150). Netmaster doesn't have a DNS server on it right now, but it will by the time the PBS Professional machines are configured.

Ubuntu doesn't come configured with a firewall or SELinux by default, which makes setting up a PBS Professional cluster setup easier because you don't have to disable those features. There are things that Ubuntu and VMware do that you don't want, though, so do a little cleanup from the installer.

To simplify the process, assign the root user a password. You can disable it later once setup is complete. Log in with the temporary user you created during the install and change root's password.

```
sudo passwd root
```

Once root has a password, log out as the temporary user and log in as root to finish the rest.

Edit the /etc/hostname file to use its proper hostname. In this case, edit the hostname for the netmaster. You will modify the commands for each VM you install.

```
netmaster.testnet.net
```

You want static IPs, so modify the file that controls that on Ubuntu. Edit the /etc/netplan/01-netcfg.yaml file. The original file should look like the text below (but your ethernet controller might not be ens33; it's on many installs but not guaranteed).

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    ens33:
      dhcp4: yes
```

It should look like this:

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    ens33:
      dhcp4: no
      addresses: [192.168.150.150/24]
      gateway4: 192.168.150.2
      nameservers:
        addresses: [192.168.150.150,8.8.8.8]
```

Change the dhcp line to be no, then add entries for addresses, gateway4, and nameservers. Change the IPs to match your network scheme, and make sure to add yourself and an upstream IP for a DNS server. The upstream server (highlighted in red above) from your company or ISP should suffice.

Next, install and partially configure SSH.

```
apt install ssh
systemctl enable ssh
```

Configure it to allow root logins. This is useful for now because SSH is convenient when performing the rest of these steps, but console will work too.

Ubuntu doesn't set time zones for you, so you will need to do that yourself.

```
timedatectl list-timezones
timedatectl set-timezone <your time zone>
(example: timedatectl set-timezone America/Detroit)
```

To make these changes work you can restart the daemons, but it's easier to just reboot the system.

```
shutdown -r now
```

The last step is to log in and remove the temporary user. This frees up the UID for the PBS Professional test users (via NIS authentication).

```
deluser <user created during quick install>
```

Configure the Network Controller VM

Since the netmaster is going to serve as your DNS server, it's best if it knows the addresses of all the servers in the cluster, so modify the /etc/hosts file and add all the machines to it.

```
192.168.150.150 netmaster.testnet.net netmaster
192.168.150.100 pbsserv.testnet.net pbsserv
192.168.150.101 pbswrkr1.testnet.net pbswrkr1
192.168.150.102 pbswrkr2.testnet.net pbswrkr2
192.168.150.149 pbsclient1.testnet.net pbsclient1
```

Now that the prep work is done, install the packages you're going to need. You need an NIS server (ypbind), an NFS server (nfs-utils), and a DNS server (dnsmasq). Dnsmasq is relatively easy to configure and it does a good job on small networks, but any DNS server will do.

```
apt install nfs-kernel-server dnsmasq
```

This next install is in a separate command because it does something interactive. The installer will ask for your NIS domain name. This does NOT have to be your normal domain, but it can be. In this walkthrough, we used PBSTESTNIS. This will take a while to time out and continue.

```
apt install nis
# Enter the NIS domain when prompted
```

Now you're going to use autofs on the PBS Professional nodes to mount user home directories when they log in, so set up the exports of the home directories. Modify the /etc/exports file to have this:

```
/home    *(rw,sync,no_root_squash,no_subtree_check)
```

Now export the NFS mounts, start all the services, and enable them so they start on boot.

```
exportfs -rav
systemctl enable nfs-server
systemctl restart nfs-server
```

Next, set up the DNS server. Systemd's "resolved" is installed by default, so before you can set up your DNS server you need to get resolved out of the way and remove the resolv.conf so you can set it up for dnsmasq.

```
systemctl stop systemd-resolved
systemctl disable systemd-resolved
rm /etc/resolv.conf
```

Add back a new /etc/resolv.conf that points to the upstream DNS server for your network or ISP. A single line will do since the file is used by dnsmasq to look things up when it can't resolve an address through its own config.

```
nameserver <DNS server IP>
(example: nameserver 8.8.8.8)
```

Modify dnsmasq to know that it is supposed to resolve addresses in its own domain. Edit the "local=" line in /etc/dnsmasq.conf to have the proper entry.

```
local=/testnet.net/
```

You need dnsmasq to act as a caching DNS server only. Everything else needed to do that is already set up by the package installer, so all that's left is to restart the service and tell it to start on boot.

```
systemctl enable dnsmasq
systemctl restart dnsmasq
```

Once DNS is set up, set up the NIS server. In Ubuntu, you need to tell the init system that this is the master NIS server in its domain, so modify the /etc/default/nis file.

```
NISSERVER=master
```

Then modify the /etc/ypserv.securenets file to talk to your network.

```
255.0.0.0       127.0.0.0
255.255.255.0   192.168.150.0
```

At this point, you should create all the users that PBS Professional will use, as well as the user accounts you'll use for testing.

```
adduser --home /home/pbsdata --ingroup users  -u 750 pbsdata
adduser --ingroup users user01
adduser --ingroup users user02
adduser --ingroup users user03
adduser --ingroup users user04
```

For the next few steps, you need to be in the /var/yp directory.

```
cd /var/yp
```

To export the pbsdata account through NIS, modify the make file. Change these entries:

```
MINUID=750
MINGID=100
MERGE_PASSWD=true
MERGE_GROUP=true
```

Start a few services you'll need.

```
Systemctl stop nis
systemctl restart nis
# This will take a little bit of time
```

Assuming all went well, the next step is to make the database entries and ensure the system starts the daemon on boot.

```
/usr/lib/yp/ypinit -m
# Do not add any hosts, unless running secondary ypbind server
# Hit <ctrl>-D to accept the defaults
systemctl restart nis
systemctl enable nis
```

Make sure PBS Professional can talk between its nodes and servers without needing passwords. There are many ways to handle that, but in this case you can use ssh. OpenSSH is free and available on all the distros PBS Professional supports. It's encrypted, so it's moderately secure across network boundaries (if needed).

Now, make sure outgoing ssh tries to use host-based authentication and turn off error messages. Edit the /etc/ssh/ssh_config file to have these settings:

```
HostbasedAuthentication yes
EnableSSHKeysign yes
StrictHostKeyChecking
UserKnownHostsFile=/dev/null
LogLevel ERROR
```

Make sure any incoming ssh sessions also use host-based authentication. Modify the /etc/ssh/sshd_config file to have these settings (the rhosts setting is optional):

```
HostbasedAuthentication yes
IgnoreRhosts no
```

Now collect all the ssh keys and put them somewhere they can be used on the server and copied easily to the PBS Professional server and nodes. To do that, create the files in the /etc/ssh directory. You might have to modify these commands for your own network IPs. You can skip this step and come back to it later if you don't have all the other machines already installed with CentOS. They don't have to be fully installed with PBS Professional and all the daemons and programs needed. It's up to you if you want to skip to setting up the rest of the machines, then come back to this step.

```
ssh-keyscan -t ecdsa pbsserv >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa pbsserv.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa 192.168.150.100 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbsserv >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbsserv.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa 192.168.150.100 >> /etc/ssh/ssh_known_hosts

ssh-keyscan -t ecdsa pbswrkr1 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa pbswrkr1.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa 192.168.150.101 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbswrkr1 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbswrkr1.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa 192.168.150.101 >> /etc/ssh/ssh_known_hosts

ssh-keyscan -t ecdsa pbswrkr2 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa pbswrkr2.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa 192.168.150.102 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbswrkr2 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbswrkr2.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa 192.168.150.102 >> /etc/ssh/ssh_known_hosts

ssh-keyscan -t ecdsa pbsclient1 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa pbsclient1.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa 192.168.150.149 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbsclient1 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa pbsclient1.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa 192.168.150.149 >> /etc/ssh/ssh_known_hosts

ssh-keyscan -t ecdsa netmaster >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa netmaster.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t ecdsa 192.168.150.150 >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa netmaster >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa netmaster.testnet.net >> /etc/ssh/ssh_known_hosts
ssh-keyscan -t rsa 192.168.150.150 >> /etc/ssh/ssh_known_hosts

cat /etc/hosts | grep 'testnet.net' | sed -e "s: :\n:g" > /etc/ssh/shosts.equiv
```

It's too much work to scp everything around to all the machines later, so take advantage of NFS and put the files in a directory that you'll be mounting on all the other machines.

```
cp /etc/ssh/ssh_known_hosts /home/user01
cp /etc/ssh/shosts.equiv /home/user01
```

Reboot the netmaster and get ready to install the rest of the network. Make sure all the configurations are working.

```
shutdown -r now
```

Set Up the Rest of the Machines in the Cluster

Now that you have the network controller set up, the rest becomes easy. Each of the PBS Professional machines are set up the same way. This is one reason why PBS Professional is easy to deploy. The next steps should be performed for each machine in the cluster (4 times total).

Because the installer created a user, it also created that user's directory in the /home directory. You need to remove those home directories because NIS and NFS will be used to mount them from the netmaster as needed. You should "archive" the old one and create a mount point.

```
mv /home /home_orig
mkdir /home
```

Next, install all the software you need to configure for the cluster. The installer will ask for the NIS domain of your network.

```
apt install autofs nfs-common nis
```

The first daemon you will configure is autofs. You need to tell autofs what to do when it sees the OS trying to access users' home directories. To do this, we need to create two new files, /etc/auto.master.d/home.autofs and /etc/auto.home. The first is a trigger file which tells autofs what to look for and what script to run when it sees it. The second defines the mount and permissions/settings for the mount.

First, edit/etc/auto.master.d/home.autofs to have:

```
/home      /etc/auto.home --timeout 600
```

Second, edit /etc/auto.home to have:

```
*   -fstype=nfs,soft,intr,rsize=8192,wsize=8192,nosuid,tcp netmaster:/home/&
```

Now, start and enable the autofs daemon.

```
systemctl enable autofs
systemctl restart autofs
```

Now that that's done, when the system tries to access a user's home directory, the system will create a mount point for the directory and mount it. This ensures PBS Professional will have everything it needs to execute jobs properly.

Next, make sure NIS works with the login daemon. To tie NIS in, edit the /etc/nsswitch.conf entries to look like this:

```
passwd:         compat systemd nis
group:          compat systemd nis
shadow:         compat nis
gshadow:        files nis
```

To avoid potential stalling after login, modify the /lib/systemd/system/systemd-logind.service file to comment out the IPAddressDeny variable:

```
#IPAddressDeny=any
```

Now that those changes are complete, it's safe to enable NIS.

```
systemctl enable nis
systemctl restart nis
```

Finish configuring the SSH daemon.

First, make sure outgoing ssh tries to use host-based authentication and turn off error messages. Edit the /etc/ssh/ssh_config file to have these settings:

```
HostbasedAuthentication yes
EnableSSHKeysign yes
```

```
StrictHostKeyChecking no
UserKnownHostsFile=/dev/null
LogLevel ERROR
```

Next, make sure any incoming ssh sessions also use host-based authentication. Modify the /etc/ssh/sshd_config file to have these settings (the rhosts setting is optional):

```
HostbasedAuthentication yes
IgnoreRhosts no
```

Now you'll leverage the NFS mount you created. That's why you copied the SSH files into the user01 directory. Run these commands to get the files properly placed on this machine:

```
mkdir /tmp/junk_mount
mount netmaster:/home/user01 /tmp/junk_mount
cp /tmp/junk_mount/ssh_known_hosts /etc/ssh
cp /tmp/junk_mount/shosts.equiv /etc/ssh
umount /tmp/junk_mount
rmdir /tmp/junk_mount
```

Now restart ssh, and you're done with the OS setup and ready to install PBS Professional.

```
systemctl restart sshd
```

Install PBS Professional

Everything up to this point is useful for installing both the commercial and open-source versions of PBS Professional. If you have the commercial version, continue through this section for an easy way to install the software. If you want to use the open-source version, go to https://pbspro.org to download the sources. Build instructions can be found at https://github.com/PBSPro/pbspro/blob/master/INSTALL. After installing the open-source version of PBS Professional, skip to the section on verifying the install.

Download and store the PBS Professional software on your system. To make it easy, store it in /home/user01 on the netmaster. These instructions will assume you did that. The file name looks like this: PBSPro_<release \ version>-<OS version>_<binary level>.tar.gz (Ex: PBSPro_19.2.4-CentOS7_x86_64.tar.gz). Then run this as root on the netmaster server:

```
cd /home/user01
tar -xzvf PBSPro_19.2.4-Ubuntu18_x86_64.tar.gz
```

This will extract the four RPMs that are packaged with the tar file into a subdirectory named PBSPro_19.2.4, within the /home/user01 directory.

Depending on which machine you are installing, you will need to run one of these three sets of commands. You should modify the PBSVERSION and PACKAGEVERSION to the PBS Professional package name you are installing.

For the PBS Professional server (this is pbsserv), run:

```
PBSVERSION=PBSPro_19.2.4;export PBSVERSION
PACKAGEVERSION= 19.2.4.20190830141245-1_amd64;export PACKAGEVERSION
su - user01
cp $PBSVERSION/pbspro-server_$PACKAGEVERSION.deb /tmp
exit
cd /tmp
PBS_SERVER=pbsserv;export PBS_SERVER
dpkg -i pbspro-server_$PACKAGEVERSION.deb
```

For the execute clients (pbswrkr1 and pbswrkr2), run:

```
PBSVERSION=PBSPro_19.2.4;export PBSVERSION
PACKAGEVERSION= 19.2.4.20190830141245-1_amd64;export PACKAGEVERSION
su - user01
cp $PBSVERSION/pbspro-execution_$PACKAGEVERSION.deb /tmp
exit
cd /tmp
PBS_SERVER=pbsserv;export PBS_SERVER
dpkg -i pbspro-execution_$PACKAGEVERSION.deb
```

For the PBS Professional client machine (this is pbsclient1), run:

```
PBSVERSION=PBSPro_19.2.4;export PBSVERSION
PACKAGEVERSION= 19.2.4.20190830141245-1_amd64;export PACKAGEVERSION
su - user01
cp $PBSVERSION/pbspro-client_$PACKAGEVERSION.deb /tmp
exit
cd /tmp
PBS_SERVER=pbsserv;export PBS_SERVER
dpkg -i pbspro-client_$PACKAGEVERSION.deb
```

What the commands above did is define variables that may change over time so you don't have to change the install script much if the version of PBS Professional changes. The overall script is more complex in what it does behind the scenes, but it saves a lot of time scp'ing files around. Here is a breakdown of the script:

```
su - user01
cp $PBSVERSION/pbspro-client_$PACKAGEVERSION.deb /tmpZ
exit
```

In essence, while installing on different machines, you become a user that is authenticated on all those machines using NIS (user01). Since you become that user with the su command, the OS knows to mount the user's home directory into /home/user01 (because of autofs). Since you extracted the PBS Professional tarball into the component RPM files in the /home/user01 directory, you now have access to the RPM files to copy them to a place from which you can install them. Then, you exit being that user and become root again.

In the next step:

```
cd /tmp
PBS_SERVER=pbsserv;export PBS_SERVER
dpkg -i pbspro-client_$PACKAGEVERSION.deb
```

You changed directory to where the user01 user copied the files. The installer likes to have PBS_SERVER as an environment variable. If it has it, it will modify the /etc/pbs.conf file that it creates to automatically have $PBS_SERVER in the PBS_SERVER entry. (It will install without that, but you will then need to go back and edit it manually to make PBS Professional work.) This saves time later. In the case of the MoM, it will also modify the "$clienthost" entry in /var/spool/pbs/mom_priv/config to have the name of the PBS Professional server in it. Then it installs the DEB file.

On the PBS Professional server machine (pbsserv) and the two client machines (pbswrkr1 & pbswrkr2), run these commands to make sure PBS Professional starts on boot:

```
systemctl enable pbs
systemctl start pbs
```

Next, tell the server what its worker nodes are. In the PBS Professional world, these are the MoM machines, or nodes. In this case, on pbsserv and logged in as root, run:

```
qmgr -c 'create node pbswrkr1'
qmgr -c 'create node pbswrkr2'
```

You will need to install a license for this cluster. If you have a license server set up, you can point PBS Professional at that with:

```
qmgr -c 'set server pbs_license_info = 6200@<license server>'
```

Otherwise, you might have a license file instead. If so, run:

```
qmgr -c 'set server pbs_license_info = <path to license file>'
```

In order to allow the client machine to run jobs on the PBS Professional server using qsub, you need to disable a little bit of security. On the server (pbsserv), as root user, run:

```
qmgr -c 'set server flatuid=true'
```

There are ways to get around this but, since it's a closed network, setting flatuid to true isn't a big security issue. Refer to the PBS Professional Administrator's Guide documentation on security about what else you can do and what effect the flatuid variable has on the PBS Professional system.

Verify the Install

Test it all out by running a job on the new cluster. First, test from the server itself. Run this:

```
su - user01
echo 'hostname' | qsub
1.pbsserv <- Output from qsub
cat STDIN.o1
pbswrkr1.testnet.net <- Output from cat command
```

As long as you see that, it's working correctly. You can run the same commands on the client with similar results. The output is formatted as <job ID>.<pbs server hostname>. The job ID will increment as more jobs are executed on the cluster. The name returned in the output file will be different too, based on which node ran the command.