# Building a PBS Professional Virtual Test Cluster with CentOS or SLES

Derrick Hendricks
Technical Specialist – Altair PBS Works™

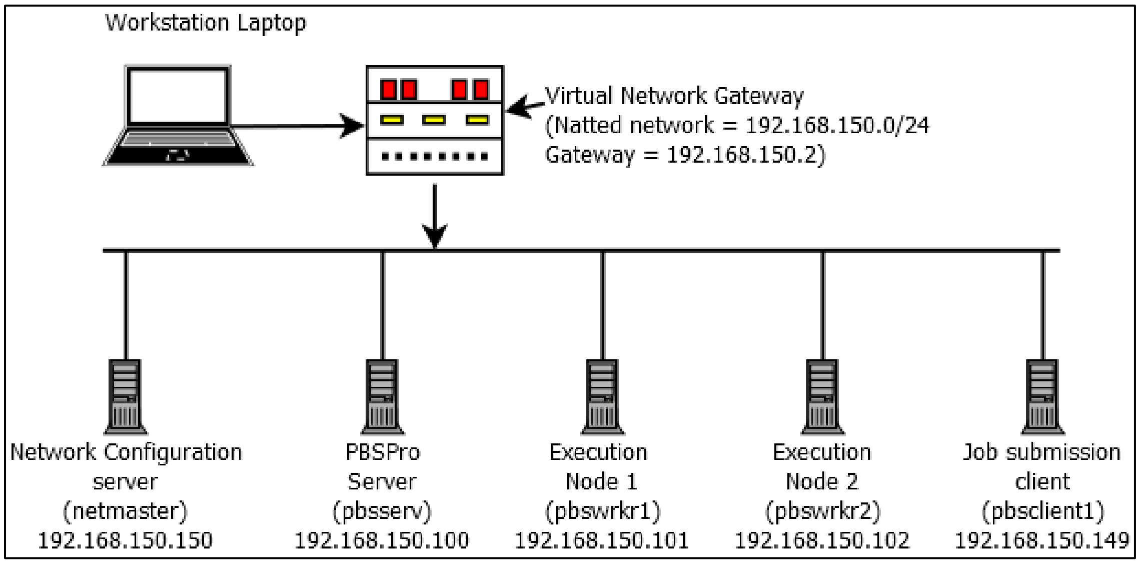
## Your PBS Professional Test Platform

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 CentOS or SUSE Linux Enterprise Server (SLES). Any VM software that can create a virtual network should be fine. For the purpose of these instructions we used VMware. CentOS was the Linux distribution we chose for the servers and nodes because it's free and it's directly supported by PBS Professional. Since SLES is relatively similar to CentOS in package management and package availability, these instructions should also apply to installing PBS Professional on that distro. Individual system-level configurations may be slightly different, but the concepts should transfer to SLES. Separate instructions are available to show you how to use Ubuntu LTS 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 the OS, we used the CentOS Minimal Install. You don't need much, and you don't want X-Windows on a compute machine. It might be good to have it installed on your network controller (netmaster) machine if you want access to some nice GUI tools for configuring the network layout.


Workstation Laptop
Virtual Network Gateway
(Natted network = 192.168.150.0/24
Gateway = 192.168.150.2)
Network Configuration server (netmaster) 192.168.150.150
PBSPro Server (pbsserv) 192.168.150.100
Execution Node 1 (pbswrkr1) 192.168.150.101
Execution Node 2 (pbswrkr2) 192.168.150.102
Job submission client (pbsclient1) 192.168.150.149

For the setup that we will go through, 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.

General guidelines:

- If a line ends with a "\" append the next line to the preceding one.
- The user of this document should always be logged into the VMs as the root user unless specifically stated otherwise.

## Configure the Network Controller Virtual Machine

At this point, you have five VMs set up with a basic install of CentOS 7.x. They have static IPs and hostnames according to the preceding table. In the case of the DNS server entries in the installer's network setup, we set the netmaster to resolve addresses using our provider's DNS servers. The other four machines 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.

Make sure the firewall won't get in the way of PBS Professional doing what it wants to do. Since this is a closed network, NAT'ed behind a gateway, it's safe enough to turn off the firewall. If you have a different setup, refer to the PBS Professional Administrator's Guide to determine what ports you need to allow. For now, disable the firewalld.

```
systemctl stop firewalld
systemctl disable firewalld
```

PBS Professional will work with SELinux in Permissive Mode, but to keep this example simple and avoid any issues with other daemons and SELinux, it's easy to disable it.

Modify the "SELINUX=" line in the /etc/selinux/config file to say:

```
SELINUX=disabled
```

Go ahead and reboot this VM so you're sure SELinux is turned off.

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.

```
yum install nfs-utils ypbind ypserv dnsmasq
```

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 rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap
systemctl start rpcbind
systemctl start nfs-server
systemctl start nfs-lock
systemctl start nfs-idmap
```

The next step is to set up your test users and the daemons to distribute them to the rest of the machines in the cluster. This is where NIS comes in. If you know enough about AD to make it work on Linux machines or want to set up this part using LDAP, that's fine. PBS Professional doesn't care how users are defined. As long as the directories all have the right permissions, the UIDs match the usernames on all the machines in the cluster, and the users can log in via SSH without passwords, PBS Professional doesn't get involved.

For this purpose, it's easy to set up NIS (ypbind).

First, set the ypdomain and make sure to use that on reboot.

```
ypdomainname PBSTESTNIS
echo "NISDOMAIN=PBSTESTNIS" >> /etc/sysconfig/network
```

The first command sets the current yp domain name and the second makes sure it is registered on boot. The next step is performed because it's good practice, and it will stop ypbind from responding to anyone else. Modify the /var/yp/securenets file to have this:

```
255.0.0.0       127.0.0.0
255.255.255.0   192.168.150.0
```

Start a few services you'll need.

```
systemctl start rpcbind ypserv ypxfrd yppasswdd
systemctl enable rpcbind ypserv ypxfrd yppasswdd
```

Initialize the yp database. The next command will start an interactive setup of yp. You don't want to add any secondary ypbind servers. Answer "y" to the "Is this correct?" question.

```
/usr/lib64/yp/ypinit -m
# Do not add any hosts, unless running secondary ypbind servers
# Answer "y" that it is correct
```

Create the users. One is a special user for PBS Professional itself. This is the one that PBS Professional uses to access its database. It needs to be a system user (with a UID less than 1000). Technically, it doesn't have to be any particular UID, and it doesn't need to be a user that is exported, but it might be useful at some point, so assign a UID and make it exportable.

```
adduser -d /home/pbsdata -g users -m -u 750 pbsdata
passwd pbsdata
```

Add a few users as test accounts. Use a password that's easy to remember. There's no need to make it difficult since this is an internal network.

```
adduser -g users user01
adduser -g users user02
adduser -g users user03
adduser -g users user04
passwd user01
passwd user02
passwd user03
passwd user04
```

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

```
cd /var/yp
```

In order to export the pbsdata account through NIS, modify the make file. Change the MINUID AND MINGID entries to this:

```
MINUID=750
MINGID=750
```

Run the make command and make sure that there are no errors.

```
make
```

Assuming all went well, enable the NIS daemons and restart them.

```
systemctl start rpcbind ypserv ypxfrd yppasswdd
systemctl enable rpcbind ypserv ypxfrd yppasswdd
```

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 no
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
```

After this next step, the network controller server (netmaster) will be completely installed and ready to go. The last thing to configure is DNS which, in this case, means dnsmasq. Edit the /etc/resolv.conf file so DNS points to your dnsmasq instance. In this case, modify it so this line comes before any other nameserver lines.

```
nameserver 127.0.0.1
```

Now configure and enable the DNS server you are now pointing to. In the/etc/dnsmasq.conf, modify the "local=" line for your domain.

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

Start and enable the new DNS server:

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

Reboot the netmaster and get ready to install the rest of the network.

```
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. We were able to install and configure all the machines in our virtual test cluster in slightly more than an hour. If the process was automated, it would take only minutes. The next steps should be performed for each machine in the cluster (4 times total).

Many of these steps are just tying in what you've already done on the netmaster server to the rest of the system.

First, disable the firewall:

```
systemctl stop firewalld
systemctl disable firewalld
```

Now, install the yum packages we are going to need to tie everything together into a stable network.

```
yum install autofs nfs-utils ypbind
```

You're installing the autofs and NFS packages because you'll need to mount the user home directories on all the machines on the cluster as needed, but you don't need to have them mounted all the time. This ensures the home environment on each machine is essentially the same. The ypbind package serves as a centralized authentication system for user accounts, so all the UIDs, groups, and permissions are the same on all the machines in the cluster. This is important for PBS Professional.

The first priority now is to get NFS working, or nothing else will function normally.

```
systemctl enable nfslock
systemctl restart nfslock
```

Make sure that autofs is configured, running, and enabled to start on boot.

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.

You'll need to set up authentication so the users you created on the netmaster can also authenticate on this server. Run these commands (changing the PBSTESTNIS entry to whatever you have set your NIS network to):

```
echo 'NISDOMAIN=PBSTESTNIS' >> /etc/sysconfig/network
echo 'domain PBSTESTNIS server netmaster.testnet.net' >> /etc/yp.conf
authconfig --enablenis --nisdomain=PBSTESTNIS --nisserver=netmaster.testnet.net \
--disablemkhomedir --update
```

Make sure those needed daemons start on boot.

```
systemctl enable ypbind
systemctl restart ypbind
systemctl enable rpcbind
systemctl restart rpcbind
```

As mentioned before, you'll need to disable SELinux, so modify the "SELINUX=" line in the /etc/selinux/config file to say:

```
SELINUX=disabled
```

Now, follow the same steps for SSH as you did on netmaster:

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
```

You're done with the OS setup and ready to install PBS Professional. Restart the VM to have the changes take effect.

```
Shutdown -r now
```

## 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-CentOS7_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:

```
su - user01
PBSVERSION=PBSPro_19.2.4;export PBSVERSION
PACKAGEVERSION=19.2.4.20190830141245-0.el7.x86_64;export PACKAGEVERSION
cp $PBSVERSION/pbspro-server-$PACKAGEVERSION.rpm /tmp
exit
cd /tmp
PBS_SERVER=pbsserv;export PBS_SERVER
rpm -i /tmp/pbspro-*.rpm
```

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

```
su - user01
PBSVERSION=PBSPro_19.2.4;export PBSVERSION
PACKAGEVERSION=19.2.4.20190830141245-0.el7.x86_64;export PACKAGEVERSION
cp $PBSVERSION/pbspro-execution-$PACKAGEVERSION.rpm /tmp
exit
cd /tmp
```

```
PBS_SERVER=pbsserv;export PBS_SERVER
rpm -i pbspro-*.rpm
```

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

```
su - user01
PBSVERSION=PBSPro_19.2.4;export PBSVERSION
PACKAGEVERSION=19.2.4.20190830141245-0.el7.x86_64;export PACKAGEVERSION
cp $PBSVERSION/pbspro-client-$PACKAGEVERSION.rpm /tmp
exit
cd /tmp
PBS_SERVER=pbsserv;export PBS_SERVER
rpm -i pbspro-*.rpm
```

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.rpm /tmp
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
rpm -i pbspro-client-$PACKAGEVERSION.rpm
```

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 RPM file. If you like to see what it's doing, you can modify the RPM command to be 'rpm -ivvvvh'.

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:

```
PATH="$PATH:/opt/pbs/bin";export PATH
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.