Storage Node Setup

A storage node (or system as your scale) is a very important unit for an HPC cluster. The computation is often about the data it produces and keeping that data safe is important. Safe here doesn't just mean security measures, but also reliability, redundancy, and recovery. Here, since we'll be using AWS, we won't concentrate as much on the safety aspects as much as configuring the basic service for using home directories and an applications directory using the Network File System (NFS) to maintain a central POSIX interface available to the our users.

We'll be utilizing the head node as a storage node in this workshop. It can be quite advantageous to make sure the head node is not the same as the storage node, but also must be budget conscious.

Pull the latest changes from the Git repository:

```
$ cd ~/build-a-cluster && git pull
```

If you had a freshly generated image from an LCI instructor, make sure to set the hostnames again:

```
$ bash ~/build-a-cluster/scripts/hostname_setup.sh

$ exec bash

$ bash ~/build-a-cluster/scripts/fix_puppet.sh

$ sudo clush -w @compute systemctl is-active puppet
```

1. Make sure NFS services is installed:
    a. `$ sudo yum install nfs-utils`
    b. `$ sudo systemctl enable nfs.service`
    c. `$ sudo systemctl start nfs.service`
2. Configure an applications directory:
    a. `$ sudo mkdir -p /exports/apps`
    b. Append the /etc/exports file to include an export of the directory
        i. `/exports/apps 10.0.0.0/28(ro,no_root_squash,sync)`
    c. Re-export the filesystems:

i. `$ sudo exportfs -r`

3. Bind mount the applications directory to the local system
    a. Create viable point
        i. `$ sudo mkdir /apps`
    b. Append entry to the file system table (i.e., /etc/fstab)
        i. `$ sudo sh -c 'echo /exports/apps /apps none defaults,bind 0 0 >> /etc/fstab'`
        ii. `$ sudo cat /etc/fstab`
    c. Mount directory
        i. `$ sudo mount -a`
    d. Verify apps is mounted
        i. `$ mount | grep apps`
4. Export the home directory (best practices have shared /home from /exports/home):
    a. Create home exports directory
        i. `$ sudo mkdir /exports/dhome`
    b. Append the /etc/exports file to export the directory to the appropriate nodes
        i. Content:

```
/exports/dhome 10.0.0.0/28(rw,no_root_squash,sync)
```

        ii. `$ sudo vim /etc/exports`
    c. Re-export the file system
        i. `$ sudo exportfs -r`
5. Bind mount the new home file system on the head node
    a. Mount point (/dhome)
        i. `$ sudo mkdir /dhome`
    b. Append entry to file system table (follow same steps as above)

c. Mount directory (follow same steps as above)

d. Verify mount exist (follow same steps as above)

6. Mount filesystems across the cluster using clush

a. `$ sudo clush -w @compute yum install -y nfs-utils`

b. `$ sudo clush -w @compute mkdir /apps /dhome`

c. `$ sudo clush -w @compute 'echo master:/exports/apps /apps nfs defaults,noauto 0 0 >> /etc/fstab'`

d. `$ sudo clush -w @compute 'echo master:/exports/dhome /dhome nfs defaults,noauto 0 0 >> /etc/fstab'`

e. `$ sudo clush -w @compute mount /apps`

f. `$ sudo clush -w @compute mount /dhome`

7. Add a cluster user (who is not the ec2-user)

a. `$ sudo -i`

b. `# useradd -b /dhome -m lcistudent`

c. `# passwd lcistudent`

d. `# clush -bw @compute ls /dhome`

e. `exit`

f. `$ sudo su - lcistudent`

g. `$ ssh-keygen`

h. Follow the on screen instructions

i. `$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys`

j. `$ chmod 600 ~/.ssh/authorized_keys`

k. `exit`

8. Installation of the Lmod modules system on master node (using clush)

a. `$ mkdir -p ~/sw`

b. `$ cd ~/sw`

c. `$ sudo yum groupinstall -y Development\ tools`

d. `$ wget -O Lmod-7.4.tar.bz2 https://sourceforge.net/projects/lmod/files/Lmod-7.4.tar.bz2/download`

e. `$ tar xvf Lmod-7.4.tar.bz2`

f. `$ cd Lmod-7.4`

g. `$ sudo mkdir -p /apps/opt`

h. `$ sudo clush -w @all yum install -y tcl lua lua-posix lua-filesystem lua-lpeg lua-term lua-json`

i. `$ ./configure --prefix=/apps/opt --with-module-root-path=/apps/opt/lmod/mf --with-shortTime=86400 --with-siteName=LCI`

j. `$ make && sudo make install`

k. `$ sudo clush -w @all --copy ~/build-a-cluster/etc/profile.d/z0*sh --dest /etc/profile.d/`

l. `$ sudo mkdir -p /apps/opt/lmod/mf/Core/lci`

m. `$ sudo touch /apps/opt/lmod/mf/Core/lci/1.0.lua`

n. `$ exec bash`

o. `$ module avail`

9. Set up profile scripts (i.e., Puppet configuration management)
    a. See `~/build-a-cluster/etc/profile.d/z*_modules.*sh`
    b. Make sure “root” doesn’t get modules!!!
    c. Copy files from the Git repository to puppet repo
        i. `$ sudo cp -R ~/build-a-cluster/etc/puppet/modules/lmod /etc/puppet/modules/`
        ii. `$ sudo vim /etc/puppet/manifests/site.pp`
            1. Add “include lmod”
        iii. `$ sudo clush -w @compute puppet agent -t`
        iv. `$ sudo clush -w @compute ls /etc/profile.d/z0*`

v. `$ sudo clush -w @compute rm -f /etc/profile.d/z0*sh`

vi. `$ sudo clush -w @compute ls /etc/profile.d/z0*`

vii. `$ sudo clush -w @compute puppet agent -t`

10. Add modulefiles for lci and the system GCC

a. `$ sudo mkdir /apps/opt/lmod/mf/Core/gcc`

b. `$ sudo cp ~/build-a-cluster/apps/opt/lmod/mf/Core/gcc/4.8.lua /apps/opt/lmod/mf/Core/gcc/`

c. `$ module avail`

d. `$ module show gcc`

e. `$ cat /apps/opt/lmod/mf/Core/gcc/4.8.lua`