Configuring: How do I add a login node?

How do I add a login node?

Table of contents

Why have a login node?

A login node is like a regular node, but runs using a software image that is configured to take care of handling login tasks. Typically, it also has another interface connected to the external network from where the logins come in.

Configuring: How do I add a login node?

Head nodes hardly get affected by logins alone. So, unless the requirements are special, a login node should not actually be needed in the first place. However, if users want to login and do extra tasks other than simply going ahead to run jobs on the regular nodes, then it can make sense to offload the login to separate login nodes, and to custom-configure these off-head login nodes for handling the extra tasks.

What do we need to consider for login nodes?

The following items can be considered during login node configuration:

1. workload management (WLM) roles are best removed for administrative convenience so that login nodes are not used as compute nodes. While all the nodes (head, login, and regular nodes) run the WLM components, login nodes are essentially submission nodes, and are not assigned cmdaemon WLM roles. The head node WLM server manages the resources that the regular (compute) node WLM clients consume. A login node submits jobs but the WLM component it uses to do the submission is outside of CMDaemon, and is also not managed by CMDaemon. For example, for submitting jobs to Torque 4 and above, the component that you have to install on a login node is `trqauthd.`
2. login nodes typically have an external interface facing the outside world, in addition to the internal one facing the regular nodes. If so, then after configuring the external interface, configuring a firewall for that external interface is sensible.
3. the logins for users via the external interface can be load balanced across the login nodes, for example using a round-robin DNS. How to deal with login nodes that may be down should be thought about.
4. restricting ssh access to the head node to just the administrators is normally desirable, since users should now be logging into the login nodes
5. restricting ssh access to the login nodes is usually best customized to the requirements of the users using the ssh options
6. if the default image is cloned to provide a basis for login nodes, then extra software should probably be added to the login nodes for the convenience of users. For example, the default image has no X11, GUI web browser, or emacs.

Based in these considerations: 1. we will take the default image and make a login image from it, making some suitable adjustments to the cluster on the way. 2. We will follow up with configuring various WLMs across the cluster, and running a job on these WLMs.

The recipes that follow can be used as a basis for your own requirements.

1. Make a login image and adjust the cluster:

clone the image

cmgui:
In the resource tree, in `Software Images`, in `Overview` tab, clone `default-image` to `login-image`, click on `Save`

cmsh:

```
[bright60->softwareimage]% clone default-image login-image ; commit
```

Cloning the node image can take a while. In cmgui you watch the event viewer; in cmsh you watch the event messages, or (if scripting) use the --wait option when a commit is done (see http://kb.brightcomputing.com/faq/index.php?action=artikel&id=102)

It is safest to wait for the cloning to be done. It typically can take about 10 minutes, then you see a message like:

```
  Copied: /cm/images/default-image->/cm/images/login-image
```

followed by:

```
  Initial ramdisk for image login-image is being generated
```

You must wait for ramdisk generation to be done before a reboot is carried out on the node. When it is done, typically after about 3 minutes, the message is:

```
  Initial ramdisk for image login-image was regenerated successfully.
```

You can then get on with the next step.

Node category creation

We make a new node category for the image, called `login`, like this:

cmgui:
create a new Node Category (clone from "`default`"). This makes a clone with the name `default1`. Rename category `default1` to `login`. Then open the `login` category, assign

new image "`login-image`" to category by selecting it from menu in the `Settings` tab. Save. After that, click on the `Roles` tab and check the box beside `Login Role`. Save.

cmsh:

```
[bright60->category]% clone login
[bright60->category*[login*]]% set softwareimage login-image; roles;
assign login; commit
```

Add extra software to image

An aside, before we add software to the `login-image`:

Running `yum --installroot=/cm/images/login-image list installed` will list the installed packages so you can figure out if you want to try removing a package.

However, the original `default-image` on which `login-image` is based is quite lean, so the chances are high that packages listed as being in `login-image` are needed or depended upon by others. So, keep an eye on the package dependencies that yum shows you before you agree to remove a package. This will help you avoid lobotomizing the image and making it unworkable. If you are unfamiliar with this kind of stripdown, don't bother with removing packages.

Now, in order to add emacs, gnuplot, and other choices (eg, eclipse-cdt) to the software image, you can run:

```
  yum --installroot=/cm/images/login-image install gnuplot emacs
eclipse-cdt
```

The above would install about 400MB of packages to the login-image. (If installing eclipse-cdt, it may install a lot of core files in the image too as a glitch in yum. In that case, just remove the core files from the image before provisioning a node with the image, or provisioning will take longer than needed, amongst other issues.).

Remove workload manager roles

Remove all workload manager (WLM) roles from the `login` category, since the head node and regular nodes do the WLM server and client roles:

cmgui:
category->role, deselect WLM roles and save.

cmsh:

```
[bright60->category[login]->roles]% unassign sgeclient; commit
```

The `sgeclient` entry here is just an example. You should remove the roles that you have. The `list` command in the prompt level above would show you any roles assigned to the login node.

Adding login nodes to CMDaemon

If you haven't created the login nodes in CMDaemon yet, you can do it with:

cmgui:

Nodes->Create Nodes button. Start with, say, login001 and end with, say, login008, and set the category to login.

cmsh:

```
[bright60]% device add physicalnode login001

[bright60->device*[login001*]]% set category login

[bright60->device*[login001*]]% foreach --clone login001 -n login002..login008 ()

Warning: The Ethernet switch settings were not cloned, and have to be set manually
```

...

```
[bright60->device*[login001*]]% commit
```

So, here we are building 8 login nodes.

You need to define the category as suggested, ie: `login`, in the preceding, because otherwise the login node defaults to having the same attributes as a default regular node.

Configure extra interface:

Login nodes usually have an extra external interface compared with regular nodes. If the BOOTIF default interface uses eth0, then you can add the eth1 physical interface name to each login node like this:

cmgui:
Nodes->login001->Network Setup->Add.
and follow the procedure.

cmsh:

```
[bright60->device[login001]]% addinterface -n login001..login008
physical eth1 externalnet 192.168.32.1; commit
```

addinterface can work with ranges of nodes and IP addresses. See the cmsh help text or manual for more on this.

Make sure that the externalnet IP address is given a static value or assigned from the DHCP server on the externalnet (if it is a DHCP-supplied address, then the server supplying it is rarely the head node).

To use cmsh to use DHCP assignment:

```
[bright60->device*[login001*]->interfaces[eth1]]% set dhcp yes; commit
```

Set up DNS resolution for login nodes from outside the cluster

The login nodes should normally be served by DNS outside the cluster (ie not served from the head nodes). Remember to configure that. Load-balancing via DNS may be a good idea.

Set up load balancing across the login nodes

The DNS to the login nodes is normally going to be an external DNS (ie not served by the head node). A simple and convenient option to distribute login sessions across the nodes is for the administrator to modify the DNS zone file, and tell users to use a generic login name that will be distributed across the login hosts. For example, if the login nodes login001 to login008 are reachable via IP addresses 1.2.0.1 to 1.2.0.8, then cluster.zone file may have these name resolution entries:

login.cm.cluster.com IN A 1.2.0.1

login.cm.cluster.com IN A 1.2.0.2

...

login.cm.cluster.com IN A 1.2.0.8

The first time the IP address is looked up for login.cm.cluster.com, the first IP address is given, the next time the lookup is done, the next IP address is given, and so on. If a login node goes down then the system administrator should adjust the records accordingly. Adjusting these automatically makes sense, for larger numbers of login nodes.

restrict ssh access to login nodes with a firewall on the login nodes

You can restrict ssh access on the login nodes to allow access only from a restricted part of the big bad internet with custom iptable rules, or perhaps something like the iptables-based shorewall (discussed next). Remember, letting users access the net means they can access websites run by evildoers. This means the administrator has to exercise due care, because users generally do not.

To install and configure shorewall:

Shorewall is an iptables rules generator. You configure it to taste, run it, and the rules are implemented in iptables. Install shorewall to the login image with the package manager, eg:

```
[root@bright60 ~]# yum --installroot=/cm/images/login-image install
shorewall
...
Total download size: 344 k
Installed size: 1.0 M
Is this ok [y/N]:
...
Complete!
```

Then, edit some of the configuration files:

```
[root@bright60 ~]# cd /cm/images/login-image/
[root@bright60 login-image]# cd etc/shorewall/ ; vi policy interfaces
zones rules
```

The **policy** file sets the general shorewall policy across the zones:
Accept packets from the firewall, and from the loc zone (internalnet), to everywhere. Drop stuff from the net to any zone. Reject all else.

```
############################################################################
#########
#SOURCE         DEST            POLICY          LOG
LIMIT:BURST
#                                               LEVEL
fw              all             ACCEPT
loc             all             ACCEPT

all             all             REJECT          info
```

The **interfaces** file decides what is allowed on what interface. If allowing dhcp broadcasts to the login node from the internal net (on eth0), set something like:

```
############################################################################
#########
#ZONE   INTERFACE       BROADCAST       OPTIONS
loc     eth0            detect          dhcp
net     eth1            detect
```

The **zones** file defines the zones:

```
############################################################################
#########
#ZONE   TYPE            OPTIONS         IN                      OUT
#                                       OPTIONS
OPTIONS
fw      firewall
net     ipv4
loc     ipv4
```

The **rules** file is where all the final tweaks can go into a newly created NEW section.

I like my pings to work. I also want some ssh connections from trusted networks to work. I put the least critical ones last because priority matters during a DOS. And I would like cmdaemon to have its connectivity. So I may end up with something like:

```
#ACTION  SOURCE                DEST      PROTO       DEST
SOURCE     ORIGINAL     RATE            USER/
SECTION NEW
ACCEPT          all             all       icmp    8
#ACCEPT:info     net             fw        tcp     22   # SSH from
everywhere
ACCEPT:info     net:1.2.3.4/32  fw        tcp     22   # SSH from lab
ACCEPT          net:5.6.7.8/29  fw        tcp     22   # SSH from all
dorms
```

```
#rate limit this address to 2/min, burst 4/min, drop if more
# (drunk frat boys running brute force scripts over the weekends in
2012)
##ACCEPT  net:5.6.7.13    fw        tcp     22   -     -     s:2/min:4
##DROP    net:5.6.7.13    fw        tcp     22
```

Here `1.2.3.4` is an IP address that I want to be able to ssh from, and `5.6.7.8/29` is a block of addresses that should be able to ssh to the login node.

Set up the shorewall service to run:

In **cmgui**

Via the `services` tab

In **cmsh**

```
[bright60->device[login001]->services]% add shorewall
[bright60->device*[login001*]->services*[shorewall*]]% commit
```

restrict ssh access to the head node with ssh options

Make sure nobody else has their keys on the head node under /root/.ssh/authorized_keys.

Modify ssh access on head to allow only root authentication. Ie, adjust `/etc/ssh/sshd_config` with

```
AllowUsers root
```

and reboot the sshd service:

```
# service sshd restart
```

If running LDAP, then modifying `/etc/security/access.conf` is another way to restrict users.

`reboot all the login nodes so that they pick up their images and configurations properly`

Just to make sure:

```
# cmsh -c "device; pexec -n=login001..login008  reboot"
```

After we have the login nodes up and running, we should make sure they do what they are supposed to, by running a job on them with a workload manager (WLM). What follows are walkthroughs to do that. They assume no WLM is already installed.

2. Set up the workload manager and run a job

We'll cover Slurm, Torque, SGE.

Slurm setup and job submission

If you are already running Slurm, you presumably have the roles and queues assigned the way you want already. If you do not have these already configured the way you want, please set them up the way you like. The *Administrator Manual* covers this in detail.

WLM Software and environment needed on the login nodes

Don't try to assign a role for the login nodes. Just make sure the munge service is available on the login node images. A

```
yum --installroot=/cm/images/login-image install munge
```

was the suggestion earlier. However if the login image was cloned from the default image that is provided by Bright, it is already going to be on the login image. So then there is no need to yum install it like that.

Set up munge as a service on the login node.

Slurm uses munge to authenticate to the rest of the cluster when submitting jobs. So it is certainly important enough to carry out the following useful extra configuration steps for it:

1. monitor the munge service on the login nodes so that you know if it crashes.
2. set up the autostart functionality for it, so that it restarts if it crashes on the login nodes

We can do it like this:

```
[root@bright61 ~]# cmsh
[bright61]% category services default
[bright61->category[login]->services]% add munge

[bright61->category*[login*]->services*[munge*]]% commit
[bright61->category[login]->services[munge]]% set autostart yes
[bright61->category*[login*]->services*[munge*]]% set monitored yes
```

```
[bright61->category*[login*]->services*[munge*]]% commit
```

See the section on *Managing And Configuring Services* in the Admin manual for more background.

<u>munge.key file sync</u>

Make sure the /etc/munge/munge.key is the same across all nodes that use munge. For a Slurm that is already active, you can just copy the key to the new login-image. For example, on the head node, copy it over like this:

```
[root@bright61 ~]# cp /etc/munge/munge.key
/cm/images/login-image/etc/munge/munge.key
```

<u>Reboot</u>

A reboot of all the login nodes ensures their images are up to date.

```
[root@bright61 ~]# cmsh -c "device; pexec -n=login001..login008 reboot
```

<u>The Slurm submission run</u>

Make sure you have an ordinary user account to test this with (See *User Management* chapter in the *Administrator Manual* if you need to check how to add a user. Basically add a user via cmgui/cmsh and set a password).

Login as a regular user (`freddy` here) to a login node, eg `login001`. Add the Slurm and favorite MPI compiler module environments (see sections on the module environment in the Administrator and User manuals for more background):

```
[freddy@login001 ~]$ module add openmpi/gcc slurm
```

Build a sample hello world executable:

```
[freddy@login001 ~]$ mpicc hello.c -o hello
```

You can use the hello.c from the *Using MPI* chapter of the *User Manual*.

Make a job script. Here's a trivial one:

```
[freddy@login001 ~]$ cat ajs.sh
#!/bin/sh
#SBATCH -o ajs.stdout
#SBATCH --time=30     #time limit to batch job
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=1
module add shared openmpi/gcc slurm
mpirun hello
```

Run it:

```
[freddy@login001 ~]$ sbatch ajs.sh

Submitted batch job 12
[freddy@login001 ~]$ cat ajs.stdout
Hello world from process 000 out of 001, processor name node001
```

There you are - you built a login node that submitted a job using Slurm.

Torque4 setup and job submission:

We will use the Slurm submission section as the guideline, since it is pretty similar.

Make sure Torque roles and queues are already configured the way you want. If you do not have these already configured the way you want, please set them up the way you like. The *Administrator Manual* covers this in detail.

Notes

It's essential to add the submission host to /etc/hosts.equiv file to be able to submit jobs. The following error message indicates that the submission host isn't added to the /etc/hosts.equiv file:

qsub: submit error (Bad UID for job execution MSG=ruserok failed validating cmsupport/cmsupport from node001.cm.cluster)

Set up trqauthd as a service on the login node.

Set up trqauthd on the login node as a service. This is just like for munge with Slurm, just copy it to the image. Then make sure Torque knows we have submit hosts, by using `qmgr` to put these hosts in its database.

```
qmgr -c 'set server submit_hosts = login001'
qmgr -c 'set server submit_hosts += login002'
qmgr -c 'set server submit_hosts += login003'
```

...

See
http://docs.adaptivecomputing.com/torque/4-0-2/Content/topics/1-installConfig/serverConfig.htm
for details on this. `qmgr -c 'print server'` lists the configuration.

We don't have to deal with a "key" file like munge.key, the database configuration is what tells Torque to trust the login hosts.

Reboot

Reboot the login images.

The Torque submission run

As for Slurm, build a sample Hello World executable. The PBS-style job script for Torque can be something like:

```
cat tjob.sh
#!/bin/bash
#
#PBS -l walltime=1:00:00
#PBS -l nodes=1
#PBS -l mem=500mb
#PBS -j oe

# If modules are needed by the script, then source modules
environment:
. /etc/profile.d/modules.sh
# Add any modules you might require:
module add shared gcc torque  openmpi/gcc

mpirun hello
```

Run it with:

```
[freddy@login001 ~]$ qsub -q shortq tjob.sh
15.bright61.cm.cluster
[freddy@login001 ~]$ cat tjob.sh.o15
Hello world from process 000 out of 001, processor name node001
```

So we have a login node that submitted a job using Torque.

SGE setup and job submission:

This is really easy. We follow the same pattern as for Slurm submission, but with fewer things to do:

Only if you have never used SGE before on the cluster: initialize SGE, allocate roles

Initializing SGE means you wipe existing jobs and queues. So you should skip this step if you are currently running an SGE system.

Only if you have never run SGE on the cluster before, then initialize SGE and assign the sgeclient/sgeserver roles across the cluster nodes. Also make sure some queues are available.

No need to install an extra daemon like `trqauthd` or `munge` on the login node, since we have cloned it from the default node configuration, which has all the SGE parts we need.

The SGE submission run

As for Slurm, build a sample Hello World executable. The job script for SGE can be something like:

```
[freddy@log001 ~]$ cat sge.sh
#!/bin/sh
#
# Your job name
#$ -N My_Job
#
# Run job through bash shell
#$ -S /bin/bash
# If modules are needed, source modules environment:
. /etc/profile.d/modules.sh
# Add any modules you might require:
module add shared sge openmpi/gcc

mpirun hello
```

Run it with:

```
[freddy@log001 ~]$ qsub -q all.q sge.sh
Your job 34 ("My_Job") has been submitted
[freddy@login001 ~]$ cat My_Job.o15
Hello world from process 000 out of 001, processor name node001
```

So we have a login node that submitted a job using SGE.

Unique solution ID: #1029
Author: Frank Furter
Last update: 2016-12-27 20:59