Crane Cluster Manual

University of the Witwatersrand
2016

Written by:
Or Hanoch
Joshua Bruton
Meir Rosendorff

Group instructor:
Atif Muhammad

Table of Contents

About the Crane Cluster

The Crane Cluster is a computer cluster made up of one head node and many separate nodes it uses. As of writing this manual the Crane cluster consisted of the headnode and 19 nodes and is located in MSL building, second floor, research lab 2.

1. HeadNode

Hostname - crane

The headnode has 1 physical Ethernet card split logically into two cards - one for the internal cluster LAN and one for internet connectivity through the wits network. The external internet connection goes through the Wits proxy and as such gets its IP using the Wits DHCP. The internal lan connection uses a static IP.

Internet IP (as of writing this manual): 10.10.187.31
LAN IP - 10.0.0.101

2. Nodes

Hostname - crane# (where # is the number of the node)

Nodes only have a static IP for the internal LAN, they do not have access to the internet directly.

LAN IP - 10.0.0.# (where # is the number of the node)

NFS - Network File System

We use an NFS server to have a shared folder for headnode and all the other nodes.

The NFS is setup with the folder on the server being /server/folder/ and the folder on the clients being /client/folder/
In our case the NFS folder is located on headnode as the folder /craneNest. Physically this folder is located on a separate 1TB HDD mounted to headnode from sdb (see Format and Mounting HDD).

1. Installation

Run the following command on both the server and the clients:

```
yum install nfs-utils
```

2. Enable and Start

Run the following commands on both the server and the clients:

```
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
```

3. Server Configuration

Create the public folder that will be shared and give it full permissions:

```
mkdir /server/folder
chmod 777 /server/folder
```

Add the following to /etc/exports to define who has permissions to access the public folder and what their permissions are. (##CLIENT IP## must be replaced by the range of client IPs, the part in the brackets defines the types of permissions. NB the only space is between the folder path and the client ip.)

```
/server/folder/ ##CLIENT IP##(rw,sync,no_root_squash,no_all_squash)
```

Restart NFS with:

```
systemctl restart nfs-server
```

Client Configuration

Create the folder in which the data inside /server/folder will be displayed:

```
mkdir /client/folder
```

Mounting

Use the following to mount the /server/folder inside the /client/folder replacing ##server IP## with the server's IP.

```
mount -t nfs ##server IP##:/server/folder /client/folder
```

(Note if the previous command times out see firewall section and then return to this section and try again.)

In order to set it to mount on boot add the following to the /etc/fstab file replacing ##server IP## with the server's IP:

```
##server IP##:/server/folder/ /client/folder/ nfs rw,sync,hard,intr 0 0
```

In order to confirm correct mount enter the df command and you should see your folder on the list.

4. Firewall

Uncomment the following in /etc/sysconfig/nfs order to set default ports

```
MOUNTD_PORT=port
STATD_PORT=port
LOCKD_TCPPORT=port
LOCKD_UDPPORT=port
```

Use the following command for each of the ports you just uncommented in /etc/sysconfig/nfs

as well as ports 111 and 2049 in order to allow nfs through the firewall where ### is the port number)

```
firewall-cmd --permanent --add-port=###/tcp
```

After allowing all the ports through the firewall use the following command to restart the firewall:

```
firewall-cmd –reload
```

MPI using NFS

MPI is a program used to run jobs, these jobs can be run in a parallel fashion across the entire cluster.

Below, MPI will be installed and setup in the shared NFS folder /craneNest/ in the apps/ directory.

1 Installation

Create the folder for mpi in apps using:

```
mkdir /craneNest/apps/openmpi
```

Go to the download location of openmpi and run the following:

```
./configure -prefix = craneNest/apps/openmpi
make
make install
```

5. Adding PATHS

You need to add craneNest/apps/openmpi/bin and craneNest/apps/openmpi/lib to the $PATH of all the nodes on the server.

You can do this without a restart by typing the following commands:

```
export PATH=$PATH:/craneNest/apps/openmpi/bin
export PATH=$PATH:/craneNest/apps/openmpi/lib
```

However to make the change permanent you must add the above commands to /home/<user_name>/.bashrc and /etc/profile. (in our case the users are crane and root)

Adding to profile will make it activate when a user logs-in and adding it to .bashrc will run it when the computer boots.

1 Configuring and Modules

Enter the command

```
which mpirun
```

If it does not return a path perform the following:

Enter the commands:

```
module avail
```

add module PATH where PATH is the path seen at the end of the previous output.

Enter the command `which mpirun` to confirm it outputs a path.

Enter the command `mpiexec --version` and it should return the details of your mpi version if the paths are configured correctly.

6. Testing

To test MPI, a host file must first be created on which is simply a list of IP addresses or domain names that you want to run MPI on.

Run MPI with the following command:

```
mpirun -hostfile nameOfHostfile -np 1 echo Hello
```

If MPI was configured correctly this should output Hello in the terminal of the computers specified in the hostfile.

Test this on a number of the nodes to confirm the $PATH's are configured correctly everywhere.

In order to test if the processes are being divided amongst the nodes correctly run the following:

(The sleep command will put the running user to sleep for the number specified in seconds. The –np flags specifies the number of proccesors to be used)

```
mpirun -hostfile nameOfHostfile -np 1 echo sleep 30
```

Before the process is completed enter the following commands into any of the nodes specified in the host file:

```
top | grep nameOfUser
```

This should produce at least one sleep process (depending on the number of processors specified and one orted process (orted is the method MPI uses to run.)

If this works then openmpi is running correctly.

DNS Server (Domain Naming System)

A DNS Server is a server that pairs numbered IP addresses to named network addresses, and creates a domain for a network to work on.
We Used headNode as our DNS server.

1 Install

Install bind by running:

```
yum install bind bind-utils -y
```

7. Configuration

Open /etc/named.conf and once there comment out listen-on-v6 port 53 { ::1; }; and add your DNS server IP to listen-on port 53 {IP;};.

In allow-query add the IP range of your slaves. e.g. localhost; 10.0.0.0/24;. The allow-transfer command can be used to configure a secondary backup DNS, if you don't have one, comment it out.

At the bottom of your file, under the already existing zone, instantiate your forward and reverse DNS lookup zones. For example using zonename as the zone's name.

```
zone “zonename" IN {
type master;
file "forward.zonename";
allow-update { none; };
};

zone "x.x.x.in-addr.arpa" IN {              // 'x.x.x' is the first 3 octets of your DNS IP in reverse e.g. '0.0.10'.
type master;
file "reverse.zonename";
allow-update { none; };
};
```

Now you have to configure and create the aforementioned zones.

Save and exit named.conf

Open /var/named/ and create forward.zonename and reverse.zonename in the folder.

8. Forward Zone

Add the following to the forward.zonename file using the number of nodes required and replacing nameofDNS,root_zonename, IPofDNS and IPofNode with the relevant information. NB make sure to include all your nodes in this file.

```
$TTL 86400
@   IN  SOA   nameofDNS.  root.zonename. (
      2011071001  ;Serial
      3600        ;Refresh
      1800        ;Retry
      604800      ;Expire
      86400       ;Minimum TTL
)
@       IN  NS          nameofDNS.
@       IN  A           IPofDNS
@       IN  A           IPofNode1
@       IN  A           IPofNode2
nameofDNS   IN  A           IPofDNS
node1  IN  A           IPofNode1
node2  IN  A           IPofNode2
```

9. Reversing Zone

Add the following lines to the reverse.zonename file using the number of nodes required and replacing nameofDNS,root_zonename, IPofDNS and IPofNode with the relevant information. NB make sure to include all your nodes in this file.

```
$TTL 86400
@   IN  SOA     nameOfDNS. root.zonename. (
      2011071001  ;Serial
      3600        ;Refresh
      1800        ;Retry
      604800      ;Expire
      86400       ;Minimum TTL
)
@       IN  NS          nameOfDNS.
@       IN  PTR         zonename.
nameOfDNS   IN  A   iPofDNS
```

```
nameOfNode1  IN  A   iPofnode1
nameOfNode2   IN  A   iPofnode2
lastoctetofDNSIP     IN  PTR        nameOfDNS
lastocterofNode1IP     IN  PTR          nameOfNode1
lastocterofNode2IP     IN  PTR            nameOfNode2
```

10. Switch On

You can now turn on the DNS by running the following

```
systemctl enable named          (Will start named on boot)
systemctl start named           (Will start named now)
```

11.Firewall

Configure the firewall to allow DNS on port 53 with the following:

```
firewall-cmd --permanent --add-port=53/tcp
firewall-cmd --permanent --add-port=53/udp
```

Reload the firewall for the new settings to take effect with:

```
firewall-cmd --reload
```

12. Permissions

Permissions, ownership and SELinux configuration is done with the following:

```
chgrp named -R /var/named
chown -v root:named /etc/named.conf
restorecon -rv /var/named
restorecon /etc/named.conf
```

13. Checking

Run the following if there is no output continue.

```
named-checkconf /etc/named.conf
```

Zones checks replace zonename with your zone's name, if it returns OK, they are okay.

```
named-checkzone zonename /var/named/forward.zonename
named-checkzone zonename /var/named/reverse.zonename
```

Add the DNS IP to your ifcfg file so it knows who to contact. (if you do not use eno1, replace it with your internet interface.)

```
vim /etc/sysconfig/network-scripts/ifcfg-eno1
```

Add `DNS="IPofDNS"` to that `ifcfg-eno1` and then save and exit.

Go to `/etc/resolv.conf` and add:

```
search zonename
nameserver IPofDNS
```

note: You will have to add this to all the `resolv.conf` files of the nodes as well AND you can only search one zone per node.

Now restart the network and DNS.

```
systemctl restart network
reboot
```

Check your `resolv.conf` file and make sure your changes are still there. If not, add them again and use the command `chattr +i /etc/resolv.conf`

To make the file uneditable. To undo this, use `chattr -i /etc/resolv.conf.`

14. Testing

Use: nslookup zonename and you should receive the following:

```
Server:         IPofDNS
Address:        IPofDNS#53

Name:  nameofZone
Address: IPofDNS
Name:  nameofZone
Address: IPofNode1
Name:  nameofZone
Address: IPofNode2
.
.
.
```

Now ping an active node using its name (not its IP-addr) and you should get a response.

If both tests work the DNS is running.

Quota

Qouta allows you to limit the size of folders for certain users

1 Setup

To setup perform the following:

add usrquota,grpquota to the folder you want to add a quota to in /etc/fstab

For example in our case we add a quota to craneNest by editing /etc/fstab in the following way:

```
/craneNest/      home   ext3     defaults,usrquota,grpquota      1 2
```

Remount the folder to apply the settings:

```
mount -o remount the folder / reboot
```

Use the following to confirm your folder has quota enabled

```
mount | grep quota
```

Now switch quota on for that folder

```
quotaon /craneNest
```

15. Setting a Quota

Now we set the actual quota for the folder

Use the following command to set it for the user replacing USER with the user's name

```
edquota -u  USER
```

and to set it for groups, replacing GROUP with the group's name we use

```
eqquota –g GROUP
```

This will open a window that looks like the following for a user named Jack

```
Disk quotas for user jack (uid 1001):
  Filesystem                   blocks       soft       hard     inodes     soft     hard
  /dev/mapper/centos-home           0       5500       6000          0        0        0
~
~
```

By editing the first two values underneath hard and soft we can change the quota on the disk. When the user's usage reaches the soft value he we will receive a warning and when he reaches the hard value it will no longer allow him to create new files.

The inodes field allows us to limit the number of files the user can make.

edquota -t allows you to configure grace period for soft limit after which it becomes hard storage

In order to track usage the command repquota -as will give a summary of all the different user's usages and their limits

Automation of the process

This can be done for user creation.

We can automate the process by creating a template user and copying those setting to other users apon their creation. In that case we use the command where USER is the user we are creating and USERprotoype is the name of the prototype user we want to use

```
edquota USER -p USERprotoype
```

Torque

Torque is a program that allows scheduling and distribution of jobs from a headnode to separate nodes of a cluster.

1 Setup on HeadNode

Opening Ports:

Torque requires certain ports to be open for essential communication.

- For client and pbs_mom communication to pbs_server, the default port is 15001.
- For pbs_server communication to pbs_mom, the default port is 15002.
- For pbs_mom communication to pbs_mom, the default port is 15003.

```
[root]# firewall-cmd --add-port=15001/tcp --permanent
[root]# firewall-cmd --reload
```

Verify Hostname:

Make sure that the correct hostname and IP are entered correctly in `/etc/hosts` (in our case "crane")

To verify that the hostname resolves correctly, make sure that `hostname` and `hostname -f` report the correct name for the host.

Install required packages:

```
[root]# yum install libtool openssl-devel libxml2-devel boost-devel gcc gcc-c++
```

Download Torque:

Go to folder you want to save Torque files at (I chose `/root/programs`)

Download git and copy folder from git:

```
[root]# yum install git
[root]# git clone https://github.com/adaptivecomputing/torque.git -b 6.0.1 6.0.1
[root]# cd 6.0.1
```

```
[root]# ./autogen.sh
```

Install Torque:

While in the folder 6.0.1

```
[root]# ./configure
[root]# make
[root]# make install
```

Set Torque server name:

Verify that the /var/spool/torque/server_name file exists and contains the correct name of the server (in our case "crane").

can be done using:

```
[root]# echo <torque_server_hostname> > /var/spool/torque/server_name
```

Configure the trqauthd daemon to start automatically at system boot:

```
[root]# cp contrib/systemd/trqauthd.service /usr/lib/systemd/system/
[root]# systemctl enable trqauthd.service
[root]# echo /usr/local/lib > /etc/ld.so.conf.d/torque.conf
[root]# ldconfig
[root]# systemctl start trqauthd.service
```

Make sure /usr/local/bin and /usr/local/sbin are in PATH env variable:

Checking what is in PATH:

```
echo $PATH
```

Adding to PATH:

```
[root]# export PATH=/usr/local/bin/:/usr/local/sbin/:$PATH
```

Initialize serverdb by executing the torque.setup script:

in 6.0.1 folder:

```
[root]# ./torque.setup root
```

Set nodes to be used:

Add nodes to the /var/spool/torque/server_priv/nodes in the format:

```
<node_host_name> np=<number_of_cores_available>
gpus=<number_of_gpus_available_(optional)> <string_to_characterize_node_(optional)>
```

Configure pbs_server to start automatically at system boot, and then start the daemon:

```
[root]# qterm
[root]# cp contrib/systemd/pbs_server.service /usr/lib/systemd/system/
[root]# systemctl enable pbs_server.service
[root]# systemctl start pbs_server.service
```

16. Installation on Nodes

Opening Ports - done on each node:

On nodes:

```
[root]# firewall-cmd --add-port=15002-15003/tcp --permanent
[root]# firewall-cmd --reload
```

Create packages to be installed on nodes - done on headnode:

from 6.0.1 folder

```
[root]# make packages
Building ./torque-package-clients-linux-x86_64.sh ...
Building ./torque-package-mom-linux-x86_64.sh ...
Building ./torque-package-server-linux-x86_64.sh ...
Building ./torque-package-gui-linux-x86_64.sh ...
Building ./torque-package-devel-linux-x86_64.sh ...
```

The package files are self-extracting packages that can be copied and executed on your production machines. Use --help for options.

Copy packages to nodes - done from headnode:

from 6.0.1 folder

```
[root]# scp torque-package-mom-linux-x86_64.sh <mom-node>:<location_on_node>
[root]# scp torque-package-clients-linux-x86_64.sh <mom-node>:<location_on_node>
```

Copy MOM startup script to nodes - done from headnode:

```
[root]# scp contrib/systemd/pbs_mom.service <mom-node>:/usr/lib/systemd/system/
```

Install MOM packages on nodes - done on each node:

```
[root]# ssh root@<mom-node>
```

Go to location packages were copied to

```
[root]# ./torque-package-mom-linux-x86_64.sh --install
[root]# ./torque-package-clients-linux-x86_64.sh --install
[root]# ldconfig
```

Configure pbs_mom to start at system boot, and then start the daemon. - done on each node:

```
[root]# systemctl enable pbs_mom.service
[root]# systemctl start pbs_mom.service
```

NOTICE:

If you change headnodes hostname you need to correct /var/spool/torque/server_name aswel

17. Installation on Clients

If you want Torque client commands installed on hosts other than the Torque Server Host (headnode)

Copy packages to client - done from headnode:

Go to 6.0.1 folder

```
[root]# scp torque-package-clients-linux-x86_64.sh <torque-client-host>:<locatio_on_client>
```

Copy the trqauthd startup script to each Torque Client Host (nodes) - done from headnode:

```
[root]# scp contrib/systemd/trqauthd.service <torque-client-host>:/usr/lib/systemd/system/
```

Install packages on Clients (nodes) - done on each client (node):

```
[root]# ./torque-package-clients-linux-x86_64.sh --install
[root]# echo /usr/local/lib > /etc/ld.so.conf.d/torque.conf
[root]# ldconfig
```

Enable and start the trqauthd service - done on each client (node):

```
[root]# systemctl enable trqauthd.service
[root]# systemctl start trqauthd.service
```

18. Enable Torque As A Service

To have it run in background waiting for commands

from 6.0.1 folder:

On nodes:

```
> cp contrib/init.d/pbs_mom /etc/init.d/pbs_mom
> chkconfig --add pbs_mom
```

On headnode:

```
> cp contrib/init.d/pbs_server /etc/init.d/pbs_server
> chkconfig --add pbs_server
```

19. Commands

create pbs server:	./torque.setup <username> (from 6.0.1)
start/stop trqauthd (for contact with client):	service trqauthd start/stop
start pbs server:	pbs_server
stop pbs server:	qterm

start mom server:	pbs_mom
start schedualer:	pbs_sched
display server settings:	qmgr -c 'p s'

Give manager and operator permission to user (i.e. to be able to use qrun)

> qmgr

Qmgr: set server managers += crane@crane

Qmgr: set server operators += crane@crane

submit job:	qsub <job_file> -c enabled (optional for checkpointing)
status of jobs:	qstate
run job:	qrun <job_number>
status of nodes:	pbsnodes
cancel job:	qdel -m "optional message with -m" <job_number>
out job on hold:	qhold <job_number>
release job from hold:	qrls <job_number>
pause running job:	qsig -s STOP <job_number>
continue running job:	qsig -s CONT <job_number>
create checkpoint:	qchkpt <job_number>vi

Job Format

Typically, a submit script is written to hold all of the parameters of a job. These parameters could include how long a job should run (walltime), what resources are necessary to run, and what to execute. The following is an example submit file:

```
#PBS -N localBlast
#PBS -S /bin/sh
#PBS -l nodes=1:ppn=2,walltime=240:00:00
#PBS -M user@my.organization.com
#PBS -m ea
source ~/.bashrc
cd $HOME/work/dir
sh myBlast.sh -i -v
```

This submit script specifies the name of the job (localBlast), what environment to use (/bin/sh), that it needs both processors on a single node (nodes=1:ppn=2), that it will run for at most 10 days, and that Torque should email "user@my.organization.com" when the job exits or aborts. Additionally, the user specifies where and what to execute.

Below are some of the commonly used PBS options in a job script file. The options start with "#PBS."

Option Description

#PBS -N myJob Assigns a job name. The default is the name of PBS job script.

#PBS -l nodes=4:ppn=2 The number of nodes and processors per node.

#PBS -q queuename Assigns the queue your job will use.

#PBS -l walltime=01:00:00 The maximum wall-clock time during which this job can run.

#PBS -o mypath/my.out The path and file name for standard output.

#PBS -e mypath/my.err The path and file name for standard error.

#PBS -j oe Join option that merges the standard error stream with the standard output stream of the job.

#PBS -W stagein=file_list Copies the file onto the execution host before the job starts. (*)

#PBS -W stageout=file_list Copies the file from the execution host after the job completes. (*)

#PBS -m b Sends mail to the user when the job begins.

#PBS -m e Sends mail to the user when the job ends.

#PBS -m a Sends mail to the user when job aborts (with an error).

#PBS -m ba Allows a user to have more than one command with the same flag by grouping the messages together on one line, else only the last command gets executed.

#PBS -r n Indicates that a job should not rerun if it fails.

#PBS -V Exports all environment variables to the job.

#PBS -M <yourEmail@company.com> sepcifies e-mail address to send to

#PBS -S specifies enviroment

20. Testing

We keep our jobs at /craneNest/Jobs

Go to Jobs directory: cd /craneNest/Jobs

Check that node status for all nodes is free

```
pbsnodes
```

if one of the nodes has a problem ssh to it and restart the mom service: `systemctl restart pbs_mom.service`, if there is problem with ssh - fix it - Torque needs ssh to work properly.

Submit a job: `qsub hello.pbs` (or other job you have)

Check status of job: `qstat`

If a job is running it has an R under "S"

If status is Q try to forcefully run it: `qrun <job_number>`

If qrun works (you see R under job status using qstat) then there might be a problem with the schedualer. restart schedualer:

```
systemctl restart pbs_sched.service
```

When job ends look at the files in your directory, you should see two new files - test.out test.err. Make you have the correct output in test.out and that test.err doesn't have problematic errors.

Note: you can edit hello.pbs to the amount of nodes you want to use and amount of cpus on each not (ppn) in the row:

```
#PBS -l nodes=14:ppn=1,walltime=199:0:30
```

Test using different amount of nodes and ppn to see that everything is OK.

LDAP (Lightweight Directory Access Protocol)

LDAP is a user management system that allows you to create users and groups that are universal for your domain. The main system is installed on a Domain Controller and client versions are installed on each of the computers in the domain,
In our case the Domain Controller is headnode and client versions are installed on each of the nodes.

Note:
In our case our domain name is “cranezone” (we set this when we set the DNS)

1 Installation

LDAP INSTALLATION ON SERVER from:

https://www.certdepot.net/rhel7-configure-ldap-directory-service-user-connection/

Install the following packages:

```
# yum install -y openldap openldap-clients openldap-servers migrationtools
```

Generate a **LDAP** password from a secret key and save an SSHA encryption of it in `/etc/openldap/passwd` (we also saved the non-encrypted password there)

```
# slappasswd -s <admin_password> -n > /etc/openldap/passwd
```

Generate a X509 certificate valid for **365** days (enter all blanks except for Common Name):

```
# openssl req -new -x509 -nodes -out /etc/openldap/certs/cert.pem \
-keyout /etc/openldap/certs/priv.pem -days 365
Generating a 2048 bit RSA private key
.....+++
.............+++
writing new private key to '/etc/openldap/certs/priv.pem'
-----
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) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:<serverHostname>.<domain>
Email Address []:
```

Secure the content of the `/etc/openldap/certs` directory:

```
# cd /etc/openldap/certs
# chown ldap:ldap *
# chmod 600 priv.pem
```

Prepare the **LDAP** database:

```
# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
```

Generate database files (don't worry about error messages!):

```
# slaptest
53d61aab hdb_db_open: database "dc=my-domain,dc=com": db_open(/var/lib/ldap/id2entry.bdb) failed: No such file or directory (2).
53d61aab backend_startup_one (type=hdb, suffix="dc=my-domain,dc=com"): bi_db_open failed! (2)
slap_startup failed (test would succeed using the -u switch)
```

Change **LDAP** database ownership:

```
# chown ldap:ldap /var/lib/ldap/*
```

Activate the **slapd** service at boot:

```
# systemctl enable slapd
```

Start the **slapd** service:

```
# systemctl start slapd
```

Check the **LDAP** activity:

```
# netstat -lt | grep ldap
```

```
tcp        0      0 0.0.0.0:ldap            0.0.0.0:*               LISTEN
tcp6       0      0 [::]:ldap               [::]:*                  LISTEN
```

To start the configuration of the **LDAP** server, add the **cosine** & **nis LDAP** schemas:

```
# cd /etc/openldap/schema
# ldapadd -Y EXTERNAL -H ldapi:/// -D "cn=config" -f cosine.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=cosine,cn=schema,cn=config"
# ldapadd -Y EXTERNAL -H ldapi:/// -D "cn=config" -f nis.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=nis,cn=schema,cn=config"
```

Then, create the /etc/openldap/changes.ldif file and paste the following lines:
Note:

There can be as little as one “dc” and as many as needed

For domain name “wits.ac.za” you will have to put dc=wits,dc=ac,dc=za

In our example our domain is “cranezone” so we use only “dc=cranezone”

You can retrieve the SSHA password from the previously saved file /etc/openldap/passwd

```
dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=cranezone

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=Manager,dc=cranezone

dn: olcDatabase={2}hdb,cn=config
```

```
changetype: modify
replace: olcRootPW
olcRootPW: {SSHA}l8A+0c+lRcymtWulFbbc3EJ1PRZz9mGg

dn: cn=config
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/certs/cert.pem

dn: cn=config
changetype: modify
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/certs/priv.pem

dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: -1

dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read by dn.base="cn=Manager,dc=cranezone" read by * none
```

Send the new configuration to the **slapd** server:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/changes.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={2}hdb,cn=config"
modifying entry "olcDatabase={2}hdb,cn=config"
modifying entry "olcDatabase={2}hdb,cn=config"
modifying entry "cn=config"
modifying entry "cn=config"
```

```
modifying entry "cn=config"
modifying entry "olcDatabase={1}monitor,cn=config"
```

Create the /etc/openldap/base.ldif file and paste the following lines:

```
dn: dc=cranezone
dc: example
objectClass: top
objectClass: domain

dn: ou=People,dc=cranezone
ou: People
objectClass: top
objectClass: organizationalUnit

dn: ou=Group,dc=cranezone
ou: Group
objectClass: top
objectClass: organizationalUnit
```

Build the structure of the directory service:

```
# ldapadd -x -w <admin_password>  -D cn=Manager,dc=cranezone -f /etc/openldap/base.ldif
adding new entry "dc=example,dc=com"
adding new entry "ou=People,dc=example,dc=com"
adding new entry "ou=Group,dc=example,dc=com"
```

Create two users for testing:

```
# mkdir /home/guests
# useradd -d /<home_directory> ldapuser01 ldapuser01
# passwd ldapuser01
Changing password for user ldapuser01.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
# useradd -d /home/guests/ldapuser02 ldapuser02
# passwd ldapuser02
Changing password for user ldapuser02.
```

```
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

21. User Account Migration

Go to the directory for the migration of the user accounts:

```
# cd /usr/share/migrationtools
```

Edit the **migrate_common.ph** file and replace in the following lines:

```
$DEFAULT_MAIL_DOMAIN = "cranezone";
$DEFAULT_BASE = "dc=cranezone";
```

Create the current users in the directory service:

```
# grep ":10[0-9][0-9]" /etc/passwd > passwd
# ./migrate_passwd.pl passwd users.ldif
# ldapadd -x -w <admin_password> -D cn=Manager,dc=cranezone -f users.ldif
adding new entry "uid=ldapuser01,ou=People,dc=example,dc=com"
adding new entry "uid=ldapuser02,ou=People,dc=example,dc=com"
# grep ":10[0-9][0-9]" /etc/group > group
# ./migrate_group.pl group groups.ldif
# ldapadd -x -w <admin_password> -D cn=Manager,dc=cranezone  -f groups.ldif
adding new entry "cn=ldapuser01,ou=Group,dc=example,dc=com"
adding new entry "cn=ldapuser02,ou=Group,dc=example,dc=com"
```

Test the configuration with the user called **ldapuser01**:

```
# ldapsearch -x cn=ldapuser01 -b dc=cranezone
```

22. Firewall Configuration

Add a new service to the firewall (**ldap**: port **tcp 389**):

```
# firewall-cmd --permanent --add-service=ldap
```

Reload the firewall configuration:

```
# firewall-cmd --reload
```

Edit the **/etc/rsyslog.conf** file and add the following line:

```
local4.* /var/log/ldap.log
```

Restart the **rsyslog** service:

```
# systemctl restart rsyslog
```

23. LDAP Client configuration

(from https://www.certdepot.net/rhel7-configure-system-use-existing-ldap-directory-service-user-group-information/)

As the **authconfig-tui** is deprecated, to configure the **LDAP** client side, there are two available options:**nslcd** and **sssd**.
In this tutorial, the **nslcd** option will be used, see the authconfig tutorial for the **sssd** option.

Install the following packages:

```
# yum install -y openldap-clients nss-pam-ldapd
```

Then, type:
Note:
our server hostname is “crane” and our domain name is “cranezone”

```
# authconfig --enableforcelegacy --update
# authconfig --enableldap --enableldapauth --ldapserver="crane.cranezone" \
--ldapbasedn="dc=cranezone" --update
```

Note1: According to your requirements, you can need to specify the –enablemkhomedir option. This option creates a local user home directory at the first connection if none exists.
Note2: Type `# authconfig –help | grep ldap` to remember the necessary options.

Put the LDAP server certificate into the `/etc/openldap/cacerts` directory:

```
# scp root@crane.cranezone:/etc/openldap/certs/cert.pem \
/etc/openldap/cacerts/cert.pem
```

Apply the correct **SELinux** context to the certificate:

```
# restorecon /etc/openldap/cacerts/cert.pem
```

Activate the **TLS** option:

```
# authconfig --enableldaptls --update
```

Test the configuration:

```
# getent passwd ldapuser02
ldapuser02:*:1001:1001:ldapuser02:/home/guests/ldapuser02:/bin/bash
```

24. NFS server configuration

To get the home directory mounted, you need to configure a NFS server.
The **NFS** server is called **instructor.example.com** in the procedure.
Note: It's not required to have the **LDAP** server and the **NFS** server on the same machine, it's only easier.

25. Automounter Client configuration

Install the following packages:

```
# yum install -y autofs nfs-utils
```

Create a new indirect **/etc/auto.guests** map and paste the following line:

```
* -rw,nfs4 instructor.example.com:/home/guests/&
```

Add the following line at the beginning of the **/etc/auto.master** file:

```
/home/guests /etc/auto.guests
```

Start the **Automounter** daemon and enable it at boot:

```
# systemctl enable autofs && systemctl start autofs
```

Test the configuration:

```
# su - ldapuser02
```

26. Creating new user:

```
# cd /usr/share/migrationtools
# useradd -d /<home_directory> ldapuser01
# passwd ldapuser01
```

```
# grep ":10[0-9][0-9]" /etc/passwd > passwd
# ./migrate_passwd.pl passwd users.ldif
# ldapadd -x -w <admin_password> -D cn=Manager,dc=cranezone -f users.ldif
```

27. Creating new group:

```
# cd /usr/share/migrationtools
# groupadd <group_name>
# grep ":10[0-9][0-9]" /etc/group > group
# ./migrate_group.pl group groups.ldif
# ldapadd -x -w <admin_password> -D cn=Manager,dc=cranezone  -f groups.ldif
```

Wake On LAN

1 Configuration

On machines that will be woken up

```
vim /etc/sysconfig/network-scripts/ifcfg-<NIC> (eno1 in our case)
```

Add the line:

```
ETHTOOL_OPTS="wol g"
```

Get MAC address from each computer and write it down:

```
ifconfig | grep ether
```

In BIOS of each computer enable Wake-On-Lan:

-Gigabyte:

- press F2 on boot and enter BIOS
- under power Power tab Enable Wake On Lan option

-HP:

- Press F10 on boot and enter bios
- Advanced >> Power-On Options >> Change "Remote Wake up Boot source" to "Remote Server"

28. Execution

On Machine that sends wake up signal:

```
ether-wake  <MAC_ADDRESS>
example: ether-wake 38:60:77:92:5A:BC
```

29. Testing

Shutdown remote node

send ether-wake command from headnode

see that the remote node turns on.

Format and Mounting HDD

When adding a HDD (Hard Disk Drive) to a machine you first need to format it, then you mount it.

In our case we wanted /craneNest to be on a separate, large, HDD. This is meant for organization and to avoid hard drive memory shortages. /craneNest is located on /dev/sdb1 which is the single partition of a 1TB HDD.

Connect HDD to power and SATA on the computer.

1 Format

Check if the computer can see your HDD:

```
cat /proc/partitions
```

See if there is a new HDD (usually sdb) and see that its size is the same as the one you put in.

Note: sdb# are partitions on the HDD, we will remove them.

30. Remove partitions

```
fdisk /dev/<new HDD>  #example: fdisk /dev/sdb
```

If the new HDD has been previously used:

make new partition table: o

make new partition: n

choose primary partition: p

choose partition number: 1

choose default settings twice: enter, enter

write the new partition table: w

mkfs.xfs /dev/sdb1

31. Mount

Create folder to be mounted to:

```
mkdir <folder_path_and_name>
mount /dev/sdb1
```

Edit fstab so it will be mounted on boot:

```
vim /etc/fstab
```

Add the following row:

```
/dev/sdb1 /craneNest xfs defaults 0 1
```

Note

If you are remounting an old broken mount make sure to unmount the broken mount using the command:

```
umount
```

Additional Scripts

We have created some scripts for our own use, they are located in /craneNest/admin/.

/craneNest/admin is added to the PATH environment variable in the root users .bashrc on headnode and as such automatically exists in the PATH variable for root.

1 network_config

This script sets the network table to create the internal LAN. It splits the network adapter logically so that one logical network card is connected to the normal Wits network through the wits proxy and gets an IP from the wits DHCP and the other logical card works with the internal LAN with static IP settings.
It is called in /etc/profile and runs automatically when a user logs in to headnode.

32. perform

This script lets you run a command on any nodes you choose, or all the nodes, using one line. The format for it is:

```
./perform “<command_between_quotations>” -on <specify node numbers with spaces as separator, head for headnode> <all for all of the nodes (not including headnode)> <node1 - node2 for all lists between node1 and node2>
```

Example:
./perform “ll | grep hello” -on 1 2 7 - 10 13 head

Note:
This script needs to be edited when adding nodes.

33. ping_all

Pings all the nodes.

Note:
This script needs to be edited when adding nodes.

34. wits_proxy

This script sets the wits proxy settings for wits.
It is called in /etc/profile and runs automatically when a user logs in.

35. wake_cranes

This script runs the wake-on-lan command on the nodes specified. The wake-on-lan needs to be enabled in the bios on the nodes for this script to work (all nodes have been enabled as of writing this manual).

The format for it is:

```
wake_cranes<node1> <node2> ...
```

Or

```
wake_nodes all
```

Note:
This script needs to be edited when adding nodes.

36. LDAP Scripts

Inside the folder /craneNest/admin/ldap you will find the following scripts:

- **installClient** - this script installs and adds the necessary certificate needed for ldap to work on a new client (node). This should be run on any new node.
- newUser <user_name>- creates a new user in the ldap database (as well as on headnode, which is the domain controller) with a user home folder located in /craneNest/admin/userAccounts and sets the size limit on the folder (using qouta)
- newGroup <group_name> - creates a new group in the LDAP database.
- deleteUser <user_name> - deletes a user from the ldap database (as well as on headnode, which is the domain controller)
- DeleteGroup <group_name> - deletes a LDAP group