Ubuntu Supercomputing Virtual Cluster Setup Guide

Definition of repository

This repository and guide is designed to guide the setup of a Ubuntu supercomputing cluster. This cluster consists of a basic Ubuntu Server install that is combined with the MPICH3 system. This gives the cluster MPI capability. By default OpenMP libraries are included with GCC which is also installed in the process of setting up MPICH3.

Set up Head Node

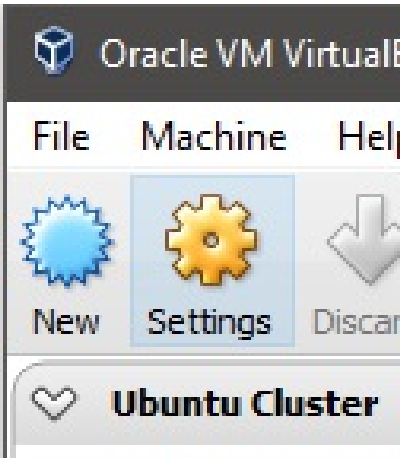
Starting Notes: You will begin by downloading the VirtualBox software. VirtualBox will allow students' to download Linux operating systems inside of a virtual environment without overwritting the user's current operating system (Windows, macOS).

> ### Step 1 - Install VirtualBox

Download and install Oracle VirtualBox
https://www.virtualbox.org/wiki/Downloads

> ### Step 2 - Create Virtual Machine

Create a virtual machine in virtualbox by starting VirtualBox and clicking the **New** button



> ### Step 3 - Create Virtual Machine Continued

Set *Name:* to **Head Node**

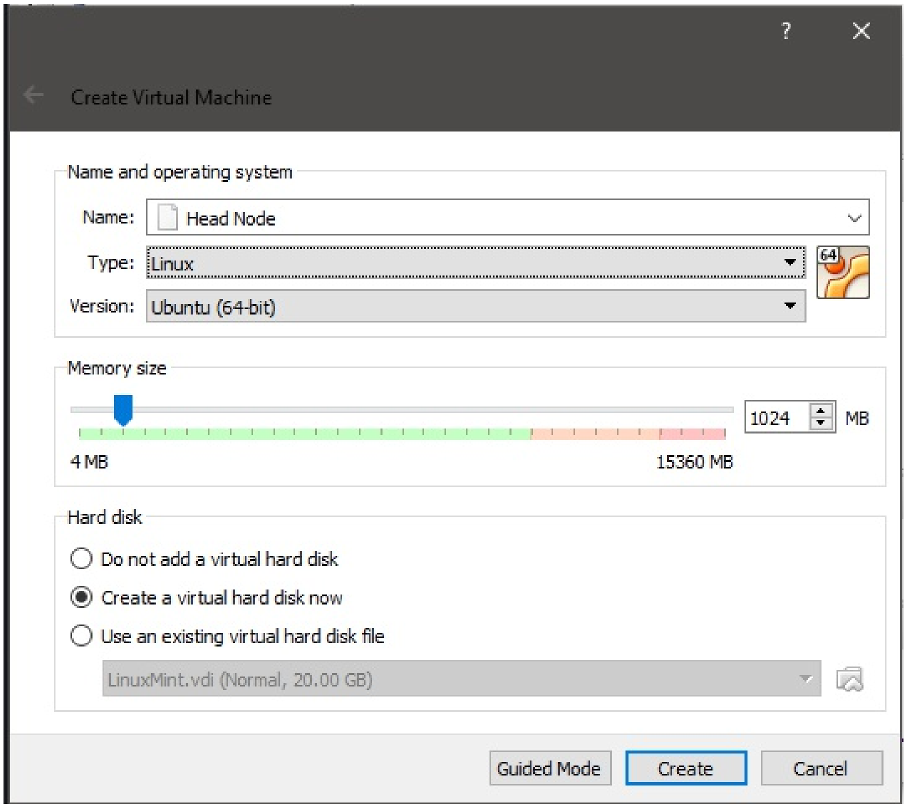
Set *Type:* to **Linux**

Set *Version:* to **Ubuntu (64-bit)**

Set *Memory size* to **2048**

Set *Hard disk* to **Create a virtual hard disk now**

Click **Create**



Create Virtual Machine

Name and operating system

Name: Head Node

Type: Linux

Version: Ubuntu (64-bit)

Memory size

1024 MB

4 MB

15360 MB

Hard disk

○ Do not add a virtual hard disk

● Create a virtual hard disk now

○ Use an existing virtual hard disk file

LinuxMint.vdi (Normal, 20.00 GB)

Step 4 - Create Virtual Hard Disk

Set *File location* to **Head Node**

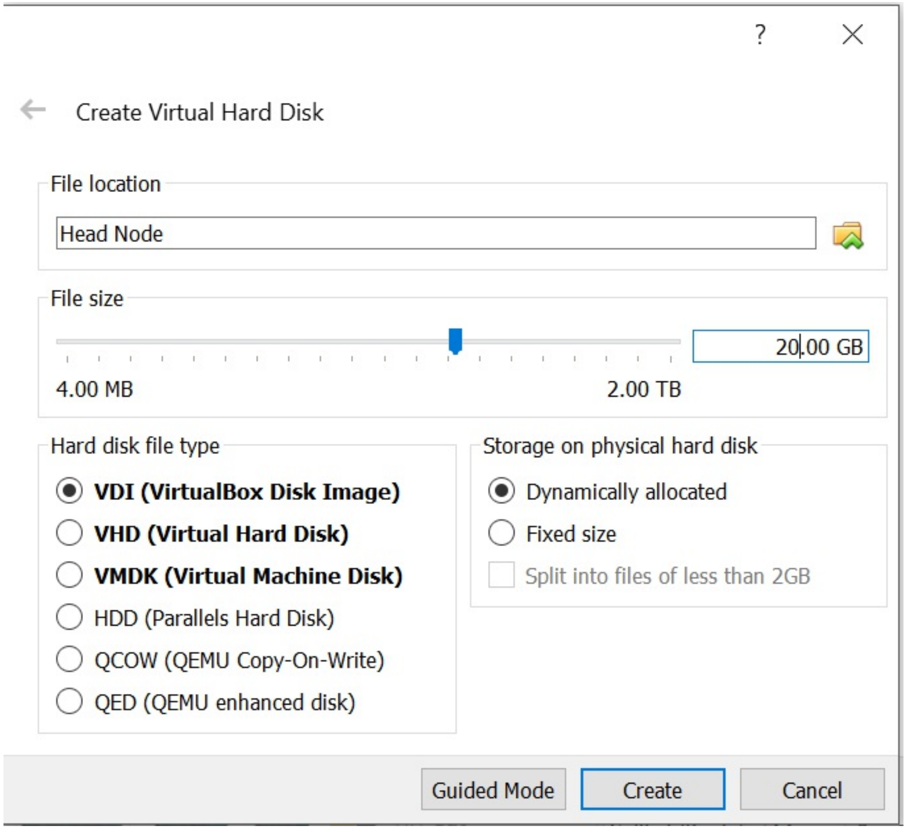
Note: File location can be changed by clicking the icon to the right of the *File location* input box

Set *File size* to **80.00**

Set *Hard disk file type* to **VDI (VirtualBox Disk Image)**

Set *Storage on physical hard disk* to **Dynamically allocated**

Click **Create**



Create Virtual Hard Disk

File location

Head Node

File size

20.00 GB

4.00 MB 2.00 TB

Hard disk file type

● VDI (VirtualBox Disk Image)

○ VHD (Virtual Hard Disk)

○ VMDK (Virtual Machine Disk)

○ HDD (Parallels Hard Disk)

○ QCOW (QEMU Copy-On-Write)

○ QED (QEMU enhanced disk)

Storage on physical hard disk

● Dynamically allocated

○ Fixed size

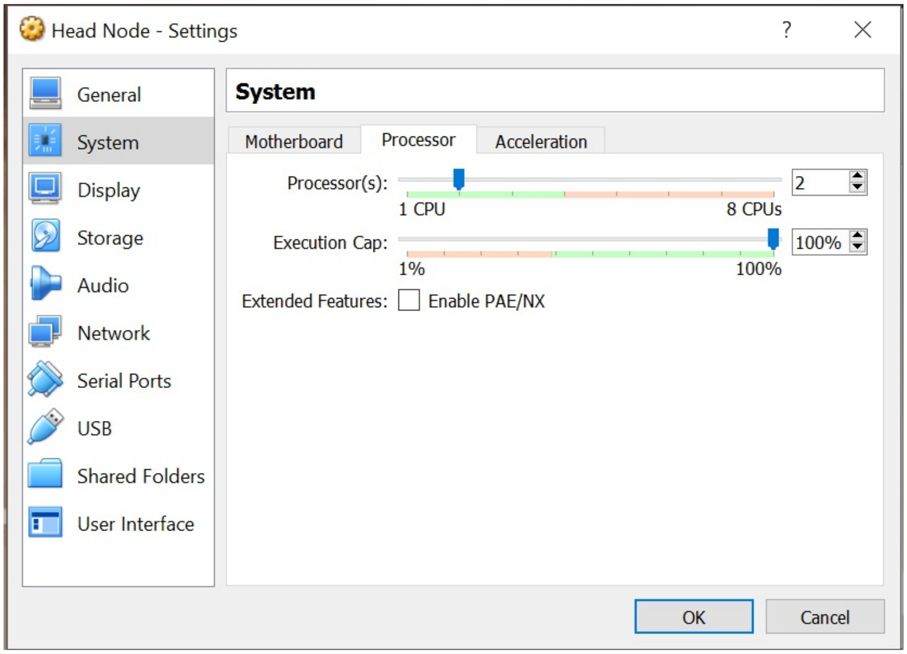
□ Split into files of less than 2GB

Step 5 - Set Processors and Network Adapters

Right click the VM for *Head Node* in the left column of VirtualBox and click on **Settings**

Click **System** and select the **Processor** tab

Change *Processor(s)* to **2**



Head Node - Settings

System

Processor(s): 2

1 CPU 8 CPUs

Execution Cap: 100%

1% 100%

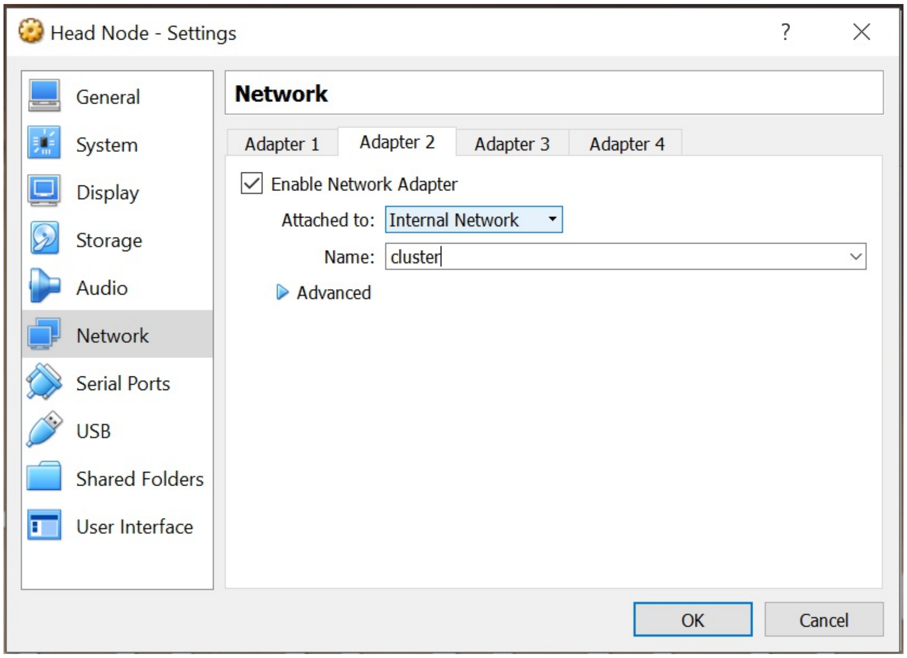
Extended Features: □ Enable PAE/NX

Click **Network** and select the **Adapter 2** tab

Check **Enable Network Adapter**

Set *Attached to:* to **Internal Network**

Set *Name:* to **cluster**



Head Node - Settings

Network

☑ Enable Network Adapter

Attached to: Internal Network

Name: cluster

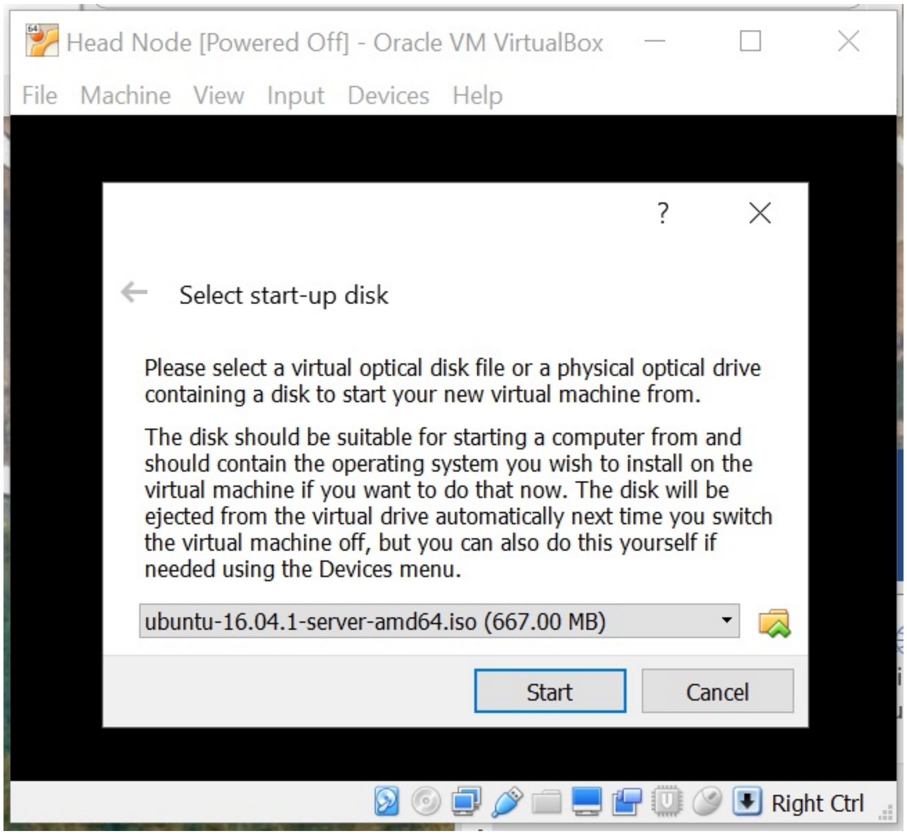
Step 6 - Start-up the Head Node VM

Download and install Ubuntu Server 64-bit ISO
https://www.ubuntu.com/download/server

Select the Head Node VM from the left column and click **Start**

Click the icon to the right of the drop-down box and navigate to the downloaded *Ubuntu Server 64-bit ISO*

Click **Start**



Head Node [Powered Off] - Oracle VM VirtualBox
File Machine View Input Devices Help
?
Start
Cancel
Right Ctrl

Select start-up disk

Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

ubuntu-16.04.1-server-amd64.iso (667.00 MB)

Step 7 - Install Ubuntu Server

Select **Install Ubuntu Server**

Select **English**

Select **United States**

Select **No** for *Detect keyboard layout*

Select **English (US)**

Select **English (US)**

Select **enp0s3** for *Primary network adapter*

Set *hostname* to **head**

Set *New user full name* to first initial and last name

Set *Username* to last name and first initial

Choose a passw ord for your account or use your student id number

Agree to use w eak passw ord

Select **No** for *Encrypt your home directory*

Select **Yes** for *Is this time zone correct?*

Select **Guided - use entire disk and set up LMV**

Select the default drive

Select **Yes** to *Write the changes to disks and configure LVM*

Keep the default drive size

Select **Yes** to *Write the changes to the disk*

Leave *HTTP Proxy* empty and Continue

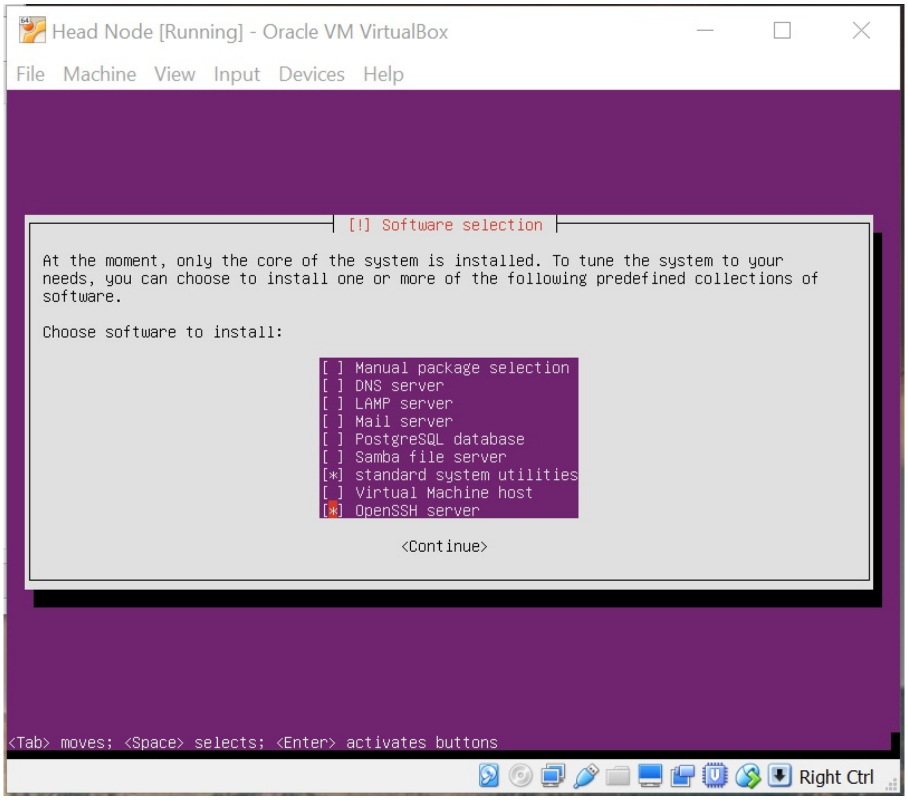
Select **No automatic updates**

Hit *Tab* to move the cursor to **OpenSSH server** and press `Space` to select it

Note: OpenSSH package is selected w hen a * is show n in the box under the cursor

Press `Enter` to continue the installation

Select **Yes** to *Install the Grub boot loader to the master boot record*



```
[!] Software selection

At the moment, only the core of the system is installed. To tune the system to your
needs, you can choose to install one or more of the following predefined collections of
software.

Choose software to install:

[ ] Manual package selection
[ ] DNS server
[ ] LAMP server
[ ] Mail server
[ ] PostgreSQL database
[ ] Samba file server
[*] standard system utilities
[ ] Virtual Machine host
[*] OpenSSH server

<Continue>

<Tab> moves; <Space> selects; <Enter> activates buttons
```

Step 8 - Set Static IP Address for Secondary Connection

Start the *Head Node* and login using the username and passw ord created during the install process

Edit the netw ork interfaces file:

```
sudo nano /etc/network/interfaces
```

Add the secondary interface to the file:

```
# Secondary Interface - cluster connection enp0s8
auto enp0s8
iface enp0s8 inet static
address 192.168.10.5
netmask 255.255.255.0
network 192.168.10.0
```

Save and exit

Edit the hosts file:

```
sudo nano /etc/hosts
```

Add to the end of the file:

```
192.168.10.5    head
192.168.10.100  node0
```

Save and exit

Step 9 - Set up IPv4 Traffic Forwarding

Enable traffic forwarding and make it permanent:

```
sudo nano /etc/sysctl.conf
```

Add the following to the end of the file:

```
# Enable IPv4 forwarding
net.ipv4.ip_forward = 1

# Disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Save and exit

Enable the new rules:

```
sudo sysctl -p
```

Enter iptables rules:

```
sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
sudo iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE
sudo bash -c "iptables-save > /etc/iptables.rules"
```

Edit */etc/network/interfaces* file:

```
sudo nano /etc/network/interfaces
```

Add the following line to the end of the file:

```
pre-up iptables-restore < /etc/iptables.rules
```

Save and exit.

Now reboot the system:

```
sudo reboot
```

Step 10 - Update the system packages and kernel

```
sudo apt udpate && sudo apt upgrade -y && sudo apt dist-upgrade -y
```

Step 11 - Set up SSH keys

VERIFY AT THE COMMAND PROMPT THAT YOU ARE UNDER YOUR USER ACCOUNT AND NOT EXECUTING CODE AS SUPER USER OR ROOT

Generate an SSH key:

```
cd ~
ssh-keygen -t rsa -C "vmcluster@swosu"
```

Press `Enter` to select default install location

Press `Enter` to leave passphrase blank

Press `Enter` to confirm blank passphrase

Copy SSH keys to authorized keys:

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

MPI

Step 1 - Create Directories

Install some required compilers and packages:

```
sudo apt-get install make build-essential gfortran
```

Create */software* directory:

```
sudo mkdir -p /software/lib/mpich_3.2
```

Create hpc user group:

```
sudo groupadd hpc
```

Add user to hpc user group:

```
sudo usermod -aG hpc <username>
```

Take ownership of */software*:

```
sudo chown -R <username>:hpc /software
```

Change to the *mpich-3.2* directory and create *build* and *install* directories:

```
cd /software/lib/mpich_3.2

mkdir build install
```

Step 2 - Download and install

Install prerequisites, download MPICH3 package and install:

```
sudo apt install gfortran
```

Download MPICH3 package:

```
wget http://www.mpich.org/static/downloads/3.2/mpich-3.2.1.tar.gz
```

Untar the package:

```
tar xvfz mpich-3.2.tar.gz
```

Change to *build* directory to begin building the install:

```
cd build
```

Configure the install:

```
/software/lib/mpich_3.2/mpich-3.2/configure  --prefix=/software/lib/mpich_3.2/install
```

Compile the install:

```
make
make install
```

Add MPI location to system environment variable PATH:

```
export PATH=$PATH:/software/lib/mpich_3.2/install/bin
```

Make the PATH change permanent by adding it to the profile file:

```
sudo nano ~/.bashrc
```

Add the follow ing to the end of the file:

```
export PATH="$PATH:/software/lib/mpich_3.2/install/bin"
```

Save and exit

Step 3 - Create Node List

Create a list of nodes for MPI to use:

```
cd ~

sudo nano nodelist
```

Save and exit.

Add the *head node* ip address to the file:

```
192.168.10.5
```

Step 4 - Test MPI

```
cd ~
```

Test 1

```
mpiexec -f nodelist hostname
```

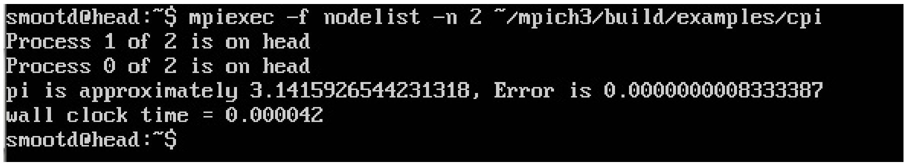
Output:

```
head
```

Test 2

```
mpiexec -f nodelist -n 2 /software/lib/mpich_3.2/build/examples/cpi
```

Output:



```
smootd@head:~$ mpiexec -f nodelist -n 2 ~/mpich3/build/examples/cpi
Process 1 of 2 is on head
Process 0 of 2 is on head
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
wall clock time = 0.000042
smootd@head:~$
```

Set up Cluster Compute Node

Step 1 - Clone the Virtual Machine

In VirtualBox right click the *Head Node* in the left column and select **Clone**

Set *Name* to **Compute Node 1**

Click **Next**

Select **Full clone**

Click **Clone**

Step 2 - Set Static IP Address

In VirtualBox select *Compute Node 1* in the left column

With *Compute Node 1* selected click **Start** in the toolbar

Login to *Compute Node 1*

Edit */etc/network/interfaces* file:

```
sudo nano /etc/network/interfaces
```

Remove all of the following lines:

```
# Secondary Interface - cluster connection enp0s8
auto enp0s8
iface enp0s8 inet static
address 192.168.10.5
netmask 255.255.255.0
network 192.168.10.0
```

Under the line `auto enp0s3` change or add the following:

```
iface enp0s3 inet static
address 192.168.10.100
netmask 255.255.255.0
gateway 192.168.10.5
dns-nameservers 8.8.8.8
```

Save and exit

Shutdown the *Compute Node 1*:

```
sudo shutdown -h now
```

Step 3 - Change Compute Node 1 Network Adapters

In VirtualBox right click *Compute Node 1* in the left column

Select **Settings**

Click **Network** and select **Adapter 2**

Uncheck the **Enable Network Adapter** box

Next, select **Adapter 1** tab

Set *Attached to:* to **Internal Network**

Set *Name:* to **cluster**

Step 4 - Set hostname

In VirtualBox select *Compute Node 1* in the left column

With *Compute Node 1* selected click **Start** in the toolbar

Login to *Compute Node 1*

Edit the hostname file:

```
sudo nano /etc/hostname
```

Change `head` to `node0`

Save and exit

Edit the hosts file:

```
sudo nano /etc/hosts
```

Change `head` to `node0`

Save and exit

Now reboot *Compute Node 1*

```
sudo reboot
```

Wait for *Compute Node 1* to reboot before continuing

Step 5 - SSH into Compute Node 1 to Acquire Authentication key

In VirtualBox select *Head Node* in the left column

With *Head Node* selected click **Start** in the toolbar

Login to *Head Node*

On *Head Node* enter:

```
ssh <username>@192.168.10.100
```

Type `yes` and press `Enter` when asked *Are you sure you want to continue connection (yes/no)?*

Type `exit` and press `Enter` to return to *Head Node*

Verify *Head Node* by checking the command prompt for `<username>@head:~$`

Step 6 - Add Compute Node 1 to the nodelist File on Head Node

On the *Head Node* edit the nodelist:

```
cd ~
sudo nano nodelist
```

Add `192.168.10.100` to the second line

Save and exit

Deploy Head Node SSH Key

Issue the following command for each node:

rsync -a --rsync-path="sudo rsync" ~/.ssh/authorized_keys pi@nodeX:~/.ssh/authorized_keys

Step 7 - Test MPI

Test 1

On the *Head Node* enter:

```
cd ~
mpiexec -f nodelist hostname
```

You should get an output similar to the following:

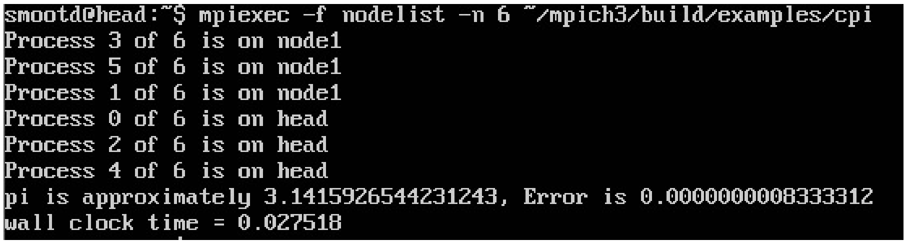
```
head
node0
```

Test 2

On the *Head Node* enter:

```
cd ~
mpiexec -f nodelist -n 6 /software/lib/mpich3/build/examples/cpi
```

You should get an output similar to the following:



```
smootd@head:~$ mpiexec -f nodelist -n 6 ~/mpich3/build/examples/cpi
Process 3 of 6 is on node1
Process 5 of 6 is on node1
Process 1 of 6 is on node1
Process 0 of 6 is on head
Process 2 of 6 is on head
Process 4 of 6 is on head
pi is approximately 3.1415926544231243, Error is 0.0000000008333312
wall clock time = 0.027518
```

Note: Each process shows which node it was executed on. You should see both head and node0 displayed. This shows that MPI is sending and executing the script on both nodes in the cluster.

Congratulations! This cluster is ready to execute MPI code.

Slurm on Head Node

Step 1 - Install needed packages

Execute:

```
sudo apt-get install slurm-wlm slurmctld slurmd
```

Step 2 - Develop configuration file

Edit */etc/slurm-llnl/slurm.conf* and add or edit to match the following:

```
# slurm.conf file generated by configurator.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
ControlMachine=head
ControlAddr=192.168.10.5
#BackupController=
#BackupAddr=
#
AuthType=auth/munge
#CheckpointType=checkpoint/none
CryptoType=crypto/munge
#DisableRootJobs=NO
#EnforcePartLimits=NO
#Epilog=
#EpilogSlurmctld=
#FirstJobId=1
#MaxJobId=999999
#GresTypes=
#GroupUpdateForce=0
#GroupUpdateTime=600
#JobCheckpointDir=/var/slurm/checkpoint
#JobCredentialPrivateKey=
#JobCredentialPublicCertificate=
#JobFileAppend=0
#JobRequeue=1
#JobSubmitPlugins=1
#KillOnBadExit=0
#LaunchType=launch/slurm
#Licenses=foo*4,bar
#MailProg=/bin/mail
#MaxJobCount=5000
#MaxStepCount=40000
#MaxTasksPerNode=128
MpiDefault=none
#MpiParams=ports=#-#
#PluginDir=
#PlugStackConfig=
#PrivateData=jobs
ProctrackType=proctrack/pgid
#Prolog=
#PrologFlags=
#PrologSlurmctld=
#PropagatePrioProcess=0
#PropagateResourceLimits=
#PropagateResourceLimitsExcept=
#RebootProgram=
ReturnToService=1
#SallocDefaultCommand=
SlurmctldPidFile=/var/run/slurm-llnl/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurm-llnl/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/lib/slurmd
SlurmUser=slurm
#SlurmdUser=root
#SrunEpilog=
#SrunProlog=
StateSaveLocation=/var/lib/slurmd/slurmctld
SwitchType=switch/none
#TaskEpilog=
TaskPlugin=task/none
#TaskPluginParam=
#TaskProlog=
#TopologyPlugin=topology/tree
#TmpFS=/tmp
#TrackWCKey=no
#TreeWidth=
#UnkillableStepProgram=
#UsePAM=0
#
#
# TIMERS
#BatchStartTimeout=10
#CompleteWait=0
#EpilogMsgTime=2000
#GetEnvTimeout=2
#HealthCheckInterval=0
#HealthCheckProgram=
InactiveLimit=0
KillWait=30
#MessageTimeout=10
#ResvOverRun=0
MinJobAge=300
#OverTimeLimit=0
SlurmctldTimeout=120
SlurmdTimeout=300
#UnkillableStepTimeout=60
#VSizeFactor=0
Waittime=0
#
#
# SCHEDULING
#DefMemPerCPU=0
FastSchedule=1
#MaxMemPerCPU=0
#SchedulerRootFilter=1
#SchedulerTimeSlice=30
SchedulerType=sched/backfill
SchedulerPort=7321
SelectType=select/linear
#SelectTypeParameters=
#
#
# JOB PRIORITY
#PriorityFlags=
#PriorityType=priority/basic
#PriorityDecayHalfLife=
#PriorityCalcPeriod=
#PriorityFavorSmall=
#PriorityMaxAge=
#PriorityUsageResetPeriod=
#PriorityWeightAge=
#PriorityWeightFairshare=
#PriorityWeightJobSize=
#PriorityWeightPartition=
#PriorityWeightQOS=
#
#
# LOGGING AND ACCOUNTING
#AccountingStorageEnforce=0
#AccountingStorageHost=
#AccountingStorageLoc=
#AccountingStoragePass=
#AccountingStoragePort=
AccountingStorageType=accounting_storage/none
#AccountingStorageUser=
AccountingStoreJobComment=YES
ClusterName=cluster
#DebugFlags=
#JobCompHost=
#JobCompLoc=
#JobCompPass=
#JobCompPort=
JobCompType=jobcomp/none
#JobCompUser=
#JobContainerType=job_container/none
JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/none
SlurmctldDebug=3
#SlurmctldLogFile=
SlurmdDebug=3
#SlurmdLogFile=
#SlurmSchedLogFile=
#SlurmSchedLogLevel=
#
#
# POWER SAVE SUPPORT FOR IDLE NODES (optional)
#SuspendProgram=
#ResumeProgram=
#SuspendTimeout=
#ResumeTimeout=
#ResumeRate=
#SuspendExcNodes=
#SuspendExcParts=
#SuspendRate=
#SuspendTime=
#
#
# COMPUTE NODES
NodeName=head CPUs=1 State=UNKNOWN
NodeName=node0 CPUs=1 State=UNKNOWN
PartitionName=vmcluster Nodes=head,node0 Default=YES MaxTime=INFINITE State=UP
```

Step 3 - Verify Slurm Controller is running

```
scontrol show daemons
```

Step 4 - Create Munge authentication keyboard

```
sudo /usr/sbin/create-munge-key
```

Step 5 - Fix Munge issue so it will load

```
sudo systemctl edit --system --full munge
```

Change this line:

```
ExecStart=/usr/sbin/munged
```

To:

```
ExecStart=/usr/sbin/munged --syslog
```

Save and exit.

Enable and start Munge:

```
sudo systemctl enable munge

sudo systemctl start munge
```

Note: The *systemctl enable munge* may show a failed notification but its fine. Just move to the next command.

Step 6 - Enable Slurm Controller

```
sudo systemctl enable slurmctld
```

Complete the automatic install:

```
sudo apt-get upgrade -y
```

Step 7 - Set create and set permissions on Slurm folder

```
sudo mkdir -p /var/lib/slurmd
sudo chown -R slurm:slurm /var/lib/slurmd
```

Reboot:

```
sudo reboot
```

Step 8 - Verify Munge and Slurm are running

```
sudo service munge status
```

Should show *Active: active (running)*

```
sudo service slurmctld status
```

Should show *Active: active (running)*

> ### Step 9 - Verify Slurm has started the partition

```
sinfo
```

Should show two entries. Look for *head* under nodelist. It's state should be *idle*. The other entry is for *node0* that we have not set up yet.

Slurm on Compute Node

> ### Step 1 - Install Slurm

On *compute node*

```
sudo apt-get install slurmd slurm-client
```

> ### Step 2 - Setup Rsync:

On *head node*

Edit the /etc/sudoers file:

```
sudo visudo
```

Add this line to the end of the file:

```
<username> ALL=NOPASSWD: /usr/bin/rsync *
```

> ### Step 3 - Copy Slurm configuration file and Munge key to *node0*:

Copy *munge.key* file:

```
sudo rsync -a --rsync-path="sudo rsync" /etc/munge/munge.key <username>@node0:/etc/munge/munge.key
```

Copy *slurm.conf* file:

```
rsync -a --rsync-path="sudo rsync" /etc/slurm-llnl/slurm.conf <username>@node0:/etc/slurm-llnl/slurm.conf
```

On *compute node*

Step 4 - Fix Munge issue so it will boot

```
sudo systemctl edit --system --full munge
```

Change this line:

```
ExecStart=/usr/sbin/munged
```

To:

```
ExecStart=/usr/sbin/munged --syslog
```

Save and exit.

Step 5 - Finish Munge setup

```
sudo systemctl enable munge

sudo systemctl start munge
```

Note: The *systemctl enable munge* may show a failed notification but its fine. Just move to the next command.

Step 6 - Enable Slurm daemon

```
sudo systemctl enable slurmd
```

Complete Slurm daemon auto install:

```
sudo apt-get upgrade -y
```

Step 7 - Set Slurm folder permissions

```
sudo chown -R slurm:slurm /var/lib/slurmd
```

Step 8 - Reboot both nodes

Execute on both nodes:

```
sudo reboot
```

Save Your Cluster Snapshot

Once your cluster is working properly you will want to take a snapshot of all nodes. This will allow you to work forward from here but to have a restore point if things don't work out with future changes.

Step 1 - Shutdown All nodes

Execute the shutdown on all nodes:

```
sudo shutdown -h now
```

Step 2 - Snapshot Your Nodes

In VirtualBox right click the node in the left column

In the upper right hand corner of VirtualBox click **Snapshots**

Click the left purple camera icon to take a snapshot of the current machine state

Give the node a name that includes its node name and stage **Example** `Head Node (MPI Stage)` or `Head Node (HADOOP/MPI Stage)`

Click **OK** and you are done

Do this for all nodes and you are safe to begin making changes and producing

Note: You can snapshot the node anywhere you want by following these instructions. In this case take advantage of the description box after naming the snapshot.

Troubleshooting

If having trouble with using rsync commands:

Setup Rsync:

On *Both nodes*

Edit the /etc/sudoers file:

```
sudo visudo
```

Add this line to the end of the file:

```
<username> ALL=NOPASSWD: /usr/bin/rsync *
```

Host Verification Key Error

In case of *host verification key* error when executing MPI follow the steps for deleting and regenerating SSH keys.

On *Head Node* as user delete previous SSH keys:

```
rm -rf ~/.ssh
mkdir ~/.ssh
```

Generate new SSH keys:

```
cd ~
ssh-keygen -t rsa -C "cluster@swosu"
```

Enter``` to leave passphrase blank

```
Copy new SSH keys nodes:
Copy SSH keys to authorized keys:
```

cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys

~~~~
```
sudo rsync -a --rsync-path="sudo rsync" ~/.ssh/authorized_keys pi@nodeX:~/.ssh/authorized_keys
~~~~

Restore VirtualBox snapshot

In VirtualBox right click the node in the left column

In the upper right hand corner of VirtualBox click **Snapshots**

Select the snapshot you wish to restore from the list

Click the second icon with the loopback green arrow to restore that snapshot

You will be prompted if you want to save a copy of the current machine state. This is a personal choice and is advised if you think you may resolve the situation causing the restore later.

Note: Remember the rule to save and save often!