Hadoop Installation Document

OS: CentOS 6.5-minimal with basic video driver.
Rationale: CentOS is a free, enterprise-class operating system with 100% binary compatibility with it's upstream source, Red Hat Enterprise Linux. RHEL is the OS used most in industry, so CentOS seemed like a great choice. With the exception of one node (so we can view Ganglia metrics, data visualizations, etc.), the minimal, non-GUI install was used in order to maximize the resources available to the Hadoop daemons.
Note: instructions cover setting up a GATEWAY IP on a dual-NIC machine so that nodes can communicate with themselves locally (192.168.1.#) as well as get out through one machine. If using a router instead of a switch, this won't be necessary.
Note #2: Hadoop clusters of even relatively moderate size can consume vast amounts of power when under full stress. Prior to setting up and benchmarking a cluster, make sure that there exists an adequate power supply to handle the cluster.

Install OS and setup 'hadoop' user:
1) Burn .iso image to USB (Win32 Disk Imager works well).
2) Escape to setup on BIOS load, specify to load/install from USB stick.
3) Follow CENTOS installation instructions.
4) Login as root and perform following:
- `useradd hadoop`
- `passwd hadoop`
- `visudo # give hadoop sudo privileges`

Post-Install:
1) Disable SELinux: set "`SELINUX=disabled`" in /etc/selinux/config
- `su`
- `echo 0 > /selinux/enforce`

2) To improve SSH speed, edit /etc/ssh/sshd_config and set all "GSSAPI" options to "no" and then restart the SSHD service:
- `sudo service sshd restart`

3) Disable IPv6 by editing /etc/sysctl.conf:
- `net.ipv6.conf.all.disable_ipv6 = 1`
- `net.ipv6.default.disable_ipv6 = 1`

4) /proc/sys/net/core/somaxconn corresponds to the limit of socket listen() backlog. On CentOS, this value defaults to 128, which is way too low to handle the bursts of requests common in a Hadoop cluster. To raise it:
- `sudo/su vi /etc/sysctl.conf` -> add net.core.somaxxconn=1024

5) By default, Linux keeps track of the last time any file was access (read, executed, etc). This means that each read access to a file will also feature a write access to update the last time it was accessed. For a Hadoop cluster, this is a performance killer. To turn it off:
- `sudo/su vi /etc/fstab` -> find the root filesystems ('/') you're using and in the fourth column should be a "defaults" value. Change this "defaults" to "defaults, noatime" and exit the file.
- `mount -o remount / #` and any other partitions on multi-disk setups

6) To reload the sysctl interface: `sudo sysctl -p`
7) Nodes within a Hadoop cluster often deal with a substantial amount of open files and require a large amount of running processes. The CentOS/Linux default is 1024, something we should increase permanently. In /etc/security/limits.conf add these lines:

```
hadoop    hard nofile    65536
hadoop    soft nofile    65536
hadoop    hard nproc     65536
hadoop    soft nproc     65536
```

8) The edits in /etc/security/limits.conf are permanent but will not take effect in the current shell. To change the values in the running shell:

- `ulimit -n 4096`
- `ulimit -u 4096`
- `ulimit -a # to ensure settings "took"`

Set up networking on dual-NIC card machine:

- ensure "NETWORKING=yes" in /etc/sysconfig/network
- edit /etc/sysconfig/network-scripts/ifcfg-eth0 and set "ONBOOT=yes" and "BOOTPROTO=dhcp"
- edit /etc/sysconfig/network-scripts/ifcfg-eth1 and set "ONBOOT=yes", "BOOTPROTO=static", "IPADDR=192.168.1.1", "NETMASK=255.255.255.0"
- `sudo service network restart`
- `sudo dhclient eth0`
- then shut down dhclient so it doesn't later wipe the static IP.

Set up IP forwarding on dual-NIC card machine:

- execute "`iptables -A FORWARD -i eth1 -j ACCEPT`" to tell firewall to allow incoming FORWARD packets over the eth1 (internal) device interface.
- execute "`iptables -A FORWARD -o eth0 -j ACCEPT`" to tell firewall to allow outgoing FORWARD packets over the eth0 (external) device interface.
- edit /etc/sysctl.conf and set "`net.ipv4.ip_forward = 1`"
- enable the above change: "`sysctl -p /etc/sysctl.conf`"
- execute "`iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`" to mask external requests from local LAN nodes with the IP address of our gateway. POSTROUTING just specifies that packets can be altered as they're leaving the firewall's networking device.
- add "`sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`" to ~/ipTemp.sh so that it doesn't have to be run every time we reboot the computer.

Set up networking on single-NIC card machines:

- edit /etc/sysconfig/network-scripts/ifcfg-eth0 and set "ONBOOT=yes", "BOOTPROTO=static", "IPADDR=192.168.1.###" and "NETMASK=255.255.255.0" (can use another identifier instead of .1. if using multiple racks - reconfigure netmask).

- edit /etc/sysconfig/network and set “NETWORKING=yes”, “GATEWAY=<internal IP address of dual-NIC card machine>”
- add DNS information to /etc/resolv.conf (should match dual-NIC’s DHCP-generated resolv.conf):
    - `search appstate.edu`
    - `nameserver 152.10.2.222`
    - `nameserver 152.10.2.223`
- `sudo service network restart`
- ensure external IP matches dual-NIC machine by doing a ‘`curl` http://myip.dnsomatic.com’
- make sure dhclient is not running else it will periodically wipe the LAN ip: “`ps -eaf | grep dhclient`”
- `sudo yum remove NetworkManager`

Set up passwordless SSH between all nodes:

- each node must generate an RSA key pair for authentication (as hadoop user): “`ssh-keygen -t rsa -f ~/.ssh/id_rsa`”
- enter passphrase that will be same for all nodes in the cluster.
- copy the generated public key back to a shared location: “`scp ~/.ssh/id_rsa.pub hadoop@192.168.1.1:~/.ssh/cluster/thisNode.pub`”
- disable StrictHostKeyChecking in /etc/ssh/ssh_config by setting “StrictHostKeyChecking no”
- “`sudo service sshd restart`”
- once all nodes have created RSA keys and copied them to a shared location, go to that location and append all public keys to the authorized_keys file: “`cat * >> ~/.ssh/authorized_keys`”
- edit the /etc/hosts file to include the IP addresses and hostnames of all machines in the cluster, as so:

```
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6
localhost6.localdomain6
192.168.1.1 aho
192.168.1.2 tito
192.168.1.3 spino
192.168.1.4 nano
192.168.1.5 ammo
192.168.1.6 techno
192.168.1.7 dryo
192.168.1.8 grypo
192.168.1.9 anono
192.168.1.10 seismo
192.168.1.11 rhino
```

```
192.168.1.12 maino
192.168.1.13 newo
192.168.1.14 drapo
192.168.1.15 mino
192.168.1.16 mino
192.168.1.17 hippo
192.168.1.18 kepo
...
```

- spread the authorized_keys file to all nodes in the cluster: "`scp ~/.ssh/authorized_keys 192.168.1.1:~/.ssh/`", etc. Once passwordless ssh is online we'll use rsync and other tools to make this sort of stuff a lot easier.
- we need to set up ssh-agent to store/provide our passphrase upon SSH attempts. To do this, copy this script I created into the home directory of the hadoop user and call it from the .bash_profile file (you can just copy over .bash_profile as well). It will handle setting up ssh-agent as well as provide an easier method to sync files across the cluster.
    - just edit NODES array to reflect the hostnames of nodes present in the cluster.

```
#!/bin/bash
#
# Hadoop Cluster start-up and admin operations
# Author:      Michael Kepple
# Called by:   ~/.bash_profile
# Date:        29 Feb 2014
#
HOSTNAME='echo $HOSTNAME | sed 's/\..*//''
SCRIPT_CONF_DIR=/home/hadoop/MastersProject/Machines/
DFS_DIRS=('/home/hdfs/' '/tmp/hdfs');
NODES=('aho' 'tito' 'spino' 'nano' 'ammo' 'techno' 'dryo' 'grypo' 'anono' 'seismo'
'rhino' 'maino' 'newo' 'appo' 'drapo' 'mino' 'hippo' 'kepo');
SLAVES=('tito' 'spino' 'nano' 'ammo' 'techno' 'dryo' 'grypo' 'anono' 'seismo' 'rhino'
'maino' 'newo' 'appo' 'drapo' 'mino' 'hippo' 'kepo');
CLASS_ACER=('nano' 'ammo' 'spino' 'techno' 'dryo' 'grypo' 'seismo' 'anono');
LOW_END=('aho' 'rhino');
MID_END=('tito' 'maino' 'drapo' 'hippo' 'kepo');
MINO=('mino');
APPO=('appo');
NEWO=('newo');
sshAgentInfo=$HOME/.ssh/agentInfo
```

```
# Run to incorporate new Datanode/Nodemanager slaves into cluster
# NOTE: run as root, argument is main node.
# ./clusterAdmin.sh -n aho
install_node()
{
    scp $1:/home/hadoop/.ssh/authorized_keys /home/hadoop/.ssh/authorized_keys
    scp $1:/etc/hosts /etc/hosts
    scp $1:/home/hadoop/.bash_profile /home/hadoop/.bash_profile
    yum install wget
    wget .
http://apt.sw.be/redhat/el6/en/x86_64/rpmforge/RPMS/rpmforge-release-0.5.3-1.el6.rf.x86_6
4.rpm
    rpm -ivh rpmforge-release-0.5.3-1.el6.rf.x86_64.rpm
    yum install ganglia ganglia-gmond
    service gmond start
    wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%24www.oracle.com"
"http://download.oracle.com/otn-pub/java/jdk/7u51-b13/jdk-7u51-linux-x64.rpm"
    rpm -ivh jdk-7u51-linux-x64.rpm
    wget -O /etc/yum.repos.d/bigtop.repo
http://www.apache.org/dist/bigtop/bigtop-0.7.0/repos/centos6/bigtop.repo
    yum install hadoop\*
    yum remove hadoop-yarn-resourcemanager
    yum remove hadoop-hdfs-secondarynamenode
    yum remove hadoop-yarn-proxyserver
    yum remove hadoop-hdfs-namenode
    yum remove hadoop-hdfs-journalnode
    rm -rf /etc/hadoop/conf
    scp -r $1:/etc/hadoop/conf /etc/hadoop/conf
    mkdir -p /home/hdfs/dfs/data
    chown -R hdfs /home/hdfs
    mkdir -p /home/hdfs/yarn
    chown -R yarn /home/hdfs/yarn
    mkdir -p /tmp/hdfs/dfs/data
    chown -R hdfs /tmp/hdfs
    yum install rsync
    yum install dmidecode
    yum install hdparm
```

```
    service hadoop-hdfs-datanode start
    service hadoop-yarn-nodemanager start
}

# Ex: sudo ./clusterAdmin.sh -d
# NOTE: Must be run as su/sudo
reformat_datanodes()
{
    stty -echo
    read -p "Password: " passw; echo
   stty echo
   for node in ${NODES[@]}
    do
       for dir in ${DFS_DIRS[@]}
       do
               sshpass -p $passw ssh $node -t "rm -f $dir/dfs/data/current/VERSION"
       done
   done
}

# Ex: ./clusterAdmin.sh -h NODES:BASIC
#     ./clusterAdmin.sh -h CLASS_ACER:BASIC
conf_sync()
{
    input=(${1//:/ })
    nodeClass=${input[0]}
    confClass=${input[1]}
    confDir=$SCRIPT_CONF_DIR$confClass/
    CLASS_NAME="${nodeClass}[@]"
    CLASS_ARRAY=( "${!CLASS_NAME}" );
    for node in ${CLASS_ARRAY[@]}
    do
       if [ "$node" = "$HOSTNAME" ]; then
           continue
       fi
       echo $node
        rsync -avz $confDir $node:/etc/hadoop/conf
    done
}
```

```
# Ex: ./clusterAdmin.sh -a
conf_sync_all()
{
    conf_sync CLASS_ACER:CLASS_ACER
    conf_sync LOW_END:LOW_END
    conf_sync MID_END:MID_END
    conf_sync MINO:MINO
    conf_sync APPO:APPO
    conf_sync NEWO:NEWO
}

# ./clusterAdmin.sh -e "hostname; ls"
execute_nodes()
{
    for node in ${SLAVES[@]}
    do
        # force pseudo-tty allocation (allows for sudo, etc).
        ssh $node -t $OPTARG
    done
}

# Should be run as root/sudo'd
# Note: master node must have installed sshpass
admin_sync()
{
    stty -echo
    read -p "Password: " passw; echo
   stty echo
   for node in ${NODES[@]}
    do
       sshpass -p $passw scp /etc/hosts $node:/etc/hosts
        sshpass -p $passw scp /home/hadoop/.ssh/authorized_keys
$node:/home/hadoop/.ssh/authorized_keys
       sshpass -p $passw ssh $node -t chown hadoop /home/hadoop/.ssh/authorized_keys
       sshpass -p $passw scp /home/hadoop/.bash_profile $node:/home/hadoop/.bash_profile
       sshpass -p $passw scp /home/hadoop/.bashrc $node:/home/hadoop/.bashrc
       sshpass -p $passw scp /home/hadoop/clusterAdmin.sh
$node:/home/hadoop/clusterAdmin.sh
```

```
        sshpass -p $passw ssh $node -t "chown hadoop:hadoop /etc/hadoop/conf/*"
        #sshpass -p $passw scp /etc/sysctl.conf $node:/etc/sysctl.conf
        #sshpass -p $passw scp /etc/security/limits.conf $node:/etc/security/limits.conf
    done
}

# Note: master node must have installed expect.
init_passphrases()
{
    for node in ${NODES[@]}
    do
        /usr/bin/expect -f ./clusterExpect $node $1
    done
}

# sudo ./clusterAdmin.sh -r
# NOTE: must be run as su/sudo
reboot()
{
    stty -echo
    read -p "Password: " passw; echo
   stty echo
   for node in ${NODES[@]}
    do
       sshpass -p $passw ssh $node -t "service hadoop-yarn-nodemanager restart"
        sshpass -p $passw ssh $node -t "service hadoop-hdfs-datanode restart"
    done
}

# ./clusterAdmin.sh -f
# NOTE: should be run on gateway node as su/sudo
gateway_forward()
{
    iptables -A FORWARD -i eth1 -j ACCEPT
    iptables -A FORWARD -o eth0 -j ACCEPT
    iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
}

while getopts "h:si:e:rn:ad" opt; do
```

```
   case $opt in
        e) execute_nodes $OPTARG
           ;;
         h) conf_sync $OPTARG
            ;;
         i) init_passphrases $OPTARG
            ;;
         s) admin_sync
           ;;
        r) reboot
           ;;
        n) install_node $OPTARG
           ;;
         a) conf_sync_all
            ;;
        d) reformat_datanodes
           ;;
    esac
done

# will exist if agent is already up - load PID, etc.
if [ -e $sshAgentInfo ]
then
   source $sshAgentInfo
fi

ssh-add -l > /dev/null
# $? indicates the error code of the last executed command - ssh-agent isn't up.
if [ $? != 0 ]
then
    # start is and store it's output to file to be sourced on other login's.
    ssh-agent -s | sed 's/^echo/#echo/' > $sshAgentInfo
    source $sshAgentInfo
    ssh-add
fi
```

- SSH is extremely strict about permissions being set correctly. From hadoop user's home directory, do a "`chmod -R 700 .ssh`", "`chmod 644 ~/.ssh/id_rsa.pub`" and "`chmod 600 ~/.ssh/id_rsa`"

- lets tell the OS to invoke clusterAdmin.sh automatically upon shell logins by adding the following to ~/.bash_profie:
    - `source clusterAdmin.sh` # should be in same dir as clusterAdmin.sh
    - make sure all nodes in the cluster get the modified .bash_profile; can SCP this from main node.

Setting up Hadoop on dual-NIC node:

- install wget:
    - `sudo yum install wget`
- Get and install current Java JDK (replace specific .rpm with current JDK file):
    - `wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%24www.oracle.com" "http://download.oracle.com/otn-pub/java/jdk/7u51-b13/jdk-7u51-linux-x64.rpm"`
- Install the JDK:
    - `sudo rpm -ivh jdk-7u51-linux-x64.rpm`
- Set JAVA_HOME environment variable in .bash_profile:
    - `JAVA_HOME=/usr/java/jdk1.7.0_51/`
- Add the Apache Bigtop repo to the list of repos YUM manages:
    - `sudo "wget -O /etc/yum.repos.d/bigtop.repo http://www.apache.org/dist/bigtop/bigtop-0.7.0/repos/centos6/bigtop.repo"`
- Go get Hadoop/etc.:
    - `sudo yum install hadoop\* flume\* mahout\* oozie\* whirr\* hbase\* hive\* hue\* pig\* sqoop\*`
- Formant the HDFS namenode:
    - `sudo /etc/init.d/hadoop-hdfs-namenode init`
- Start the HDFS daemons:
    - `sudo service hadoop-hdfs-namenode start`
    - `sudo service hadoop-hdfs-datanode start`
- Initialize some HDFS idrectories for the daemon's we'll be running:
    - `sudo /usr/lib/hadoop/libexec/init-hdfs.sh`
- The defualt /tmp directory can be limitied in size by default by the OS (CentOS included). Let's change it (/etc/hadoop/conf/core-site.xml):
    - ```
      <property>
          <name>hadoop.tmp.dir</name>
          <value>/home/hdfs</value>
      </property>
      ```
- We need to specify the directory we want YARN to write into on the HDFS filesystem (/etc/hadoop/conf/yarn-site.xml):
    - ```
      <property>
          <name>yarn.app.mapreduce.am.staging-dir</name>
          <value>/user</value>
      </property>
      ```
- Though the resourcemanager node doesn't need these values explicitly set (they'll

default to the correct ones), setting the yarn.resourcemanager.address and yarn.resourcemanager.resource-tracker.address properties will allow you to simple SCP the /etc/hadoop/conf directory to other nodes for a basic, initial cluster setup.

- CentOS automatically creates some disk partitions when it installs itself and by default limits the size of the '/' directory to 50GB, approximately. Just as we did with the /tmp directory, let's move the directory where YARN stores it's staging data and temporary map/reduce outputs:
  - `<property>`
    `<name>yarn.nodemanager.local-dirs</name>`
    `<value>/home/hadoop-yarn/cache/${user.name}/nm-local-dir`
    `</value>`
    `</property>`
- Now we create the directory we indicated above:
  - `sudo mkdir -p /home/hadoop-yarn/cache/michael/nm-local-dir`
  - `sudo chown -R yarn /home/hadoop-yarn`
  - Periodically use df to monitor partition fullness during large M/R runs. Using something like "`sudo du /<dir> | sort -n -r | head -n 100`" can help find troublesome directories.
- When executing MapReduce jobs, we need to specify that they should run on our YARN cluster and not in pseduo-distributed or local mode (/etc/hadoop/conf/mapred-site.xml):
  - `<property>`
    `<name>mapreduce.framework.name</name>`
    `<value>yarn</value>`
    `</property>`
- Now make sure that the 'hdfs' user can write to our new temp directory:
  - `mkdir ~/hdfs`
  - `mkdir ~/hdfs/dfs`
  - `mkdir ~/hdfs/dfs/data`
  - `cd`
  - `sudo chown -R hdfs hdfs`
  - make sure 'hdfs' user has read/write/execute access in all directories on the way to the new data directory.
- Start up the YARN daemons:
  - `sudo service hadoop-yarn-resourcemanager start`
  - `sudo service hadoop-yarn-nodemanager start`
- Ensure that the HDFS filesystem initialized correctly:
  - `hadoop fs -ls -R /`
- Set up HUE (edit /etc/hue/conf/hue.ini):
  - `secret_key=<30ish random characters>`
  - `in [[ mapred_clusters ]] [[[ default ]]], set "submit_to=False"`
  - `in [[ mapred_clusters ]] [[[ default ]]], set "thrift_port=9090"`

    - in [[mapred_clusters]] [[[default]]], set "hadoop_mapred_home=/usr/lib/hadoop-mapreduce"
    - in [[yarn_clusters]] [[[default]]], set "hadoop_mapred_home=/usr/lib/hadoop-mapreduce"
- goto /etc/hadoop/conf and add the following property to hdfs-site.xml:
    - <property>
        <name>dfs.webhdfs.enabled</name>
        <value>true</value>
      </property>
- in /etc/hadoop/conf/core-site.xml, add user 'hue' and 'hadoop' to the list of proxyusers/proxygroups that can impersonate the proper HDFS permissions for arbitrary accesses/commands:
    - <property>
        <name>hadoop.proxyuser.hue.hosts</name>
        <value>*</value>
      </property>
      <property>
        <name>hadoop.proxyuser.hue.groups</name>
        <value>*</value>
      </property>
    - <property>
        <name>hadoop.proxyuser.hadoop.hosts</name>
        <value>*</value>
      </property>
      <property>
        <name>hadoop.proxyuser.hadoop.groups</name>
        <value>*</value>
      </property>
- sudo groupadd supergroup
- sudo usermod -a -G supergroup hadoop
- hadoop fs -mkdir /user/hadoop
- to deal with a conflict between the default HBASE REST API port and the Nodemanager, set the HBASE port to something else in /etc/hbase/conf:
    - <property>
        <name>hbase.rest.port</name>
        <value>8070</value>
      </property>
- sudo service hbase-rest restart
    - Make sure it's running on another port now:
    - sudo netstat -tulpn | grep 8070
        - look resulting pid up with "ps -eaf | grep <pid>"
- Add all Datanode/NodeManager slave machines to /etc/hadoop/conf/slaves on the master node so that the Hadoop daemons can remotely launch/kill the slave daemons on the remote machines.
- Now restart the nodemanager, which should no longer die because of the port

conflict:
    - `sudo service hadoop-yarn-nodemanager start`
    - `sudo service hadoop-yarn-nodemanager status`
- Now let's test that MapReduce can run on our new YARN setup:
    - `hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples*.jar pi 10 1000`

Setting up Hadoop on other cluster nodes:

- install wget:
    - `sudo yum install wget`
- Get and install current Java JDK (replace specific .rpm with current JDK file):
    - `wget --no-cookies --no-check-certificate --header "Cookie: gpw_e24=http%3A%2F%24www.oracle.com" "http://download.oracle.com/otn-pub/java/jdk/7u51-b13/jdk-7u51-linux-x64.rpm"`
- Install the JDK:
    - `sudo rpm -ivh jdk-7u51-linux-x64.rpm`
- Set JAVA_HOME environment variable in .bash_profile:
    - `JAVA_HOME=/usr/java/jdk1.7.0_51/`
- Add the Apache Bigtop repo to the list of repos YUM manages:
    - "`wget -O /etc/yum.repos.d/bigtop.repo http://www.apache.org/dist/bigtop/bigtop-0.7.0/repos/centos6/bigtop.repo`"
- Go get Hadoop/etc.:
    - `sudo yum install hadoop\*`
    - `sudo yum remove hadoop-yarn-resourcemanager`
    - `sudo yum remove hadoop-hdfs-secondarynamenode`
    - `sudo yum remove hadoop-yarn-proxyserver`
    - `sudo yum remove hadoop-hdfs-namenode`
- Goto /etc/hadoop/conf and edit the following to tell Hadoop about the cluster:
    - In /etc/hadoop/conf/core-site:
        - ```
          <property>
                <name>fs.default.name</name>
          <value>hdfs://<insertNamenodeHere>:8020</value>
          </property>
          ```
- In core-site.xml, also add the permission info we added to the dual-NIC core-site as well (it's needed everywhere).
- Update YARN info, too:
    - In /etc/hadoop/conf/yarn-site.xml:
        -

```
<property>
  <name>yarn.resourcemanager.address</name>
  <value>aho:8032</value>
</property>
```
```

```
  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>aho:8031</value>
  </property>
  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>aho:8033</value>
  </property>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>aho:8030</value>
  </property>
```

- The defualt /tmp directory can be limitied in size by default by the OS (CentOS included). Let's change it (/etc/hadoop/conf/core-site.xml):
  - ```
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hdfs</value>
    </property>
    ```
- We need to specify the directory we want YARN to write into on the HDFS filesystem (/etc/hadoop/conf/yarn-site.xml):
  - ```
    <property>
        <name>yarn.app.mapreduce.am.staging-dir</name>
        <value>/user</value>
    </property>
    ```
- CentOS automatically creates some disk partitions when it installs itself and by default limits the size of the '/' directory to 50GB, approximately. Just as we did with the /tmp directory, let's move the directory where YARN stores it's staging data and temporary map/reduce outputs:
  - ```
    <property>
        <name>yarn.nodemanager.local-dirs</nam>

    <value>/home/hadoop-yarn/cache/${user.name}/nm-local-dir</valu
    e>
    </property>
    ```
- Now we created the directory we indicated above:
  - `sudo mkdir -p /home/hadoop-yarn/cache/michael/nm-local-dir`
  - `sudo chmod -R yarn /home/hadoop-yarn`
  - Periodically use df to monitor partition fullness during large M/R runs. Using something like "`sudo du /<dir> | sort -n -r | head -n 100`" can help find troublesome directories.
- When executing MapReduce jobs, we need to specify that they should run on our YARN cluster and not in pseduo-distributed or local mode (/etc/hadoop/conf/mapred-site.xml):
  - ```
    <property>
    ```

```
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
        </property>
```

- `sudo service iptables stop`
  - will otherwise encounter TCP ack errors between nodes behind gateway.
- Make sure the directories we specified are available:
  - `sudo mkdir -p /home/hdfs/dfs/data`
  - `sudo chown -R hdfs /home/hdfs`
  - `sudo mkdir /home/hdfs/yarn`
  - `sudo chown -R yarn /home/hdfs/yarn`
- To start off with, all single-card machines will share the same Hadoop configuration files (we'll optimize later). After the first six steps, new nodes can scp the /etc/hadoop/conf directory from other previously set up nodes.