JBoss EAP 5.1 Clustering

Hands on Lab

MOD_CLUSTER

Table of Contents

Introduction

Overview

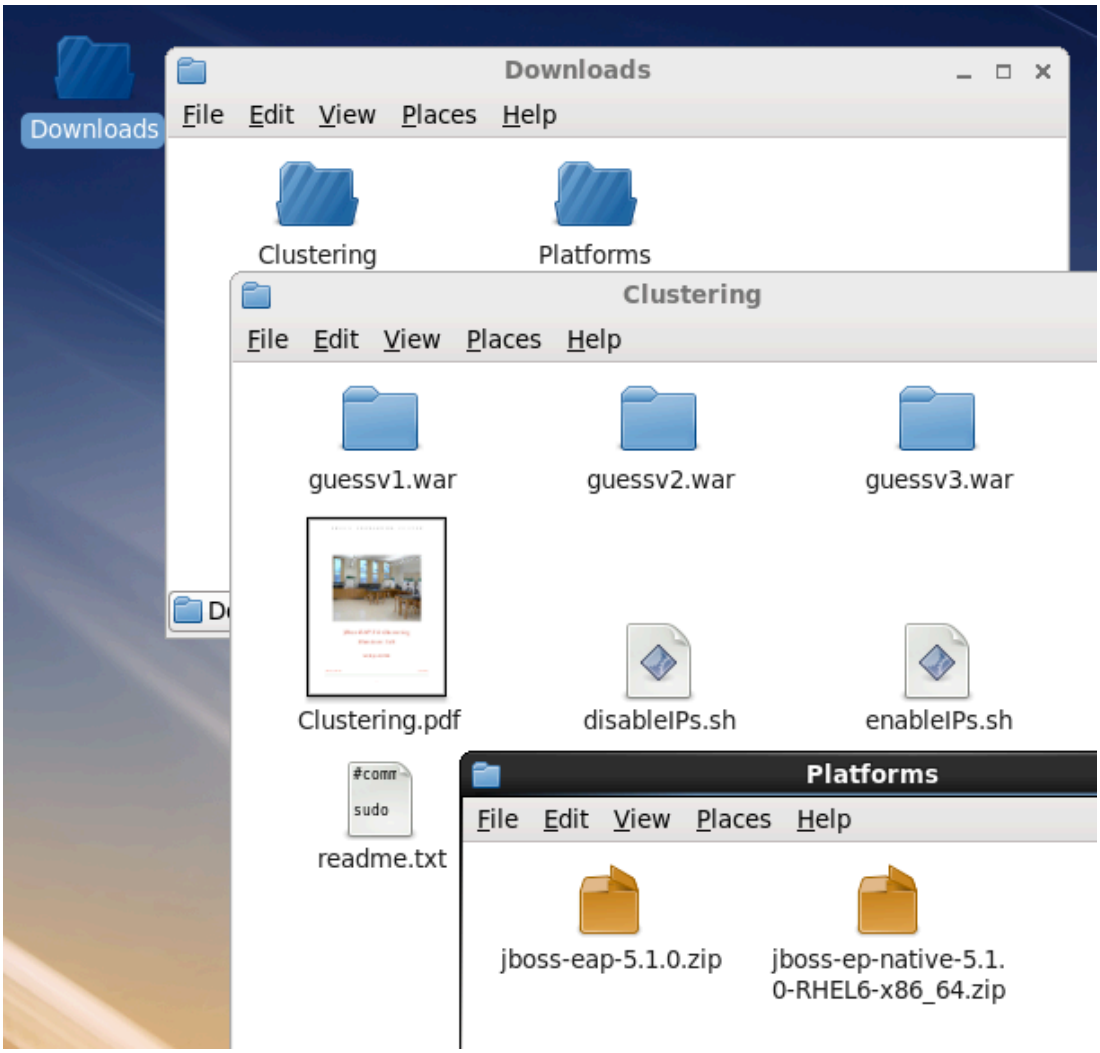
JBoss clustering is a simple way to give your website more availability and capacity for your end users. This is done simply be creating/installing several instances of JBoss and then load balancing them via some sort of proxy. In this lab we will use Apache HTTPD as the port 80 web server. We will then install mod_cluster to dynamically discover the nodes that are able to handle the load. An appendix will walk you through using/configuring SE Linux, however, the main flow of the lab will turn off SE Linux. The scope of SE Linux is far outside of the scope of this lab, but is included via the appendix to get you thinking about SE Linux implications. Once everything is installed we will explore turning on and off various instances, application deployment, rolling updates etc.

This lab is written specifically for Red Hat Enterprise Linux 6.0 (RHEL), and should work with minor path changes for future versions of RHEL. For the JBoss components, this lab was created with EAP 5.1 and EAP 5.1 native components. As future versions of these are released this lab should work with minor path changes reflecting those updated binaries. For other operating supported operating systems the below steps should essentially be unchanged.

Included FIles

If you are getting this lab as part of a Red Hat delivered lab, all of the required files should be on the Desktop in various folders. If you are downloading this the recommended structure ${Desktop}/Downloads/Platforms for your two JBoss files Application Platform 5.1.0 Binary, and JBoss EAP 5.1. Native Components for RHEL 6, x86_64. It would be recommended to have this guide in a ${Desktop}/Downloads/Clustering along with the sample war files, two scripts for enabling IP Address, and a readme.txt that has the various commands available for cutting and pasting.

You can see this structure below:



System Expectations

It is expected that you have a computer with RHEL 6.0. It is expected you will have the environment PATH set to include a JDK 6.0 to use for these labs. It is also a good idea to have JAVA_HOME set to your JDK that you plan on using. If you are in a Red Hat provided lab these settings are already done for you. Please make sure you do this before running any of the labs. Two examples of what these settings might look like is below:

```
PATH=${Some Path}/jdk1.6.0_20/bin:${Some Path}/ant/apache-ant-1.8.1: ${More Path Info}
JAVA_HOME=${Some Path}jdk1.6.0_20
```

To verify that this is correct you will have to look at these values on your system One simple way to check the JDK version that you have is to run:

```
java -version
```

to see which one is in your path, and it should be a JDK 6 version to run this lab.

Please note if you are using your own computer having an existing CLASSPATH environment variable set may cause odd issues with jar class loading, it is recommended to have this empty and not set. Please make sure to back up this value for when the lab is over. You are welcome to not do this, however weird things may happen when you are running through the labs if you do not have an empty CLASSPATH variable.

What is Expected of You

This lab is intended for self directed study, and is being delivered as a courtesy to our customers, if you are in a Red Hat led lab, please feel free to raise your hands with any questions that you have about the lab; feel free to ask why it is you are doing something, or if something does not feel right. If you are having issues, for other users forums should be available to assist you with any questions. Please know that all care was made in creating this user guide, but all screen shots and steps along the way might be off by just a little so please be patient with any issues, and feel free to raise them in the forums, or at jira.jboss.com/cookbooks

Check List

Check List

Sometimes you just need a quick list of the steps to do something, as it is something you do every so often, but you are not sure of all of the steps. If you need complete handholding, that is what the below lab chapters deliver, however if you know the gist of what you need to do, this check list is provided to help you get going.

1. Get Required Software
2. Unzip EAP
3. Unzip Native Components
4. Copy all to node1
5. Make configuration file changes to node1
6. Make sure node1 starts
7. copy node1 to node2, node3, node4
8. Make sure apache starts
9. install mod_cluster components
10. Make sure apache still starts
11. verify installation

Lab Number 1: Install and Configure Clustering

Get the File

In the ${USER_HOME}Downloads/Platforms directory you will find the EAP installer, it platform agnostic and it should look something like this:

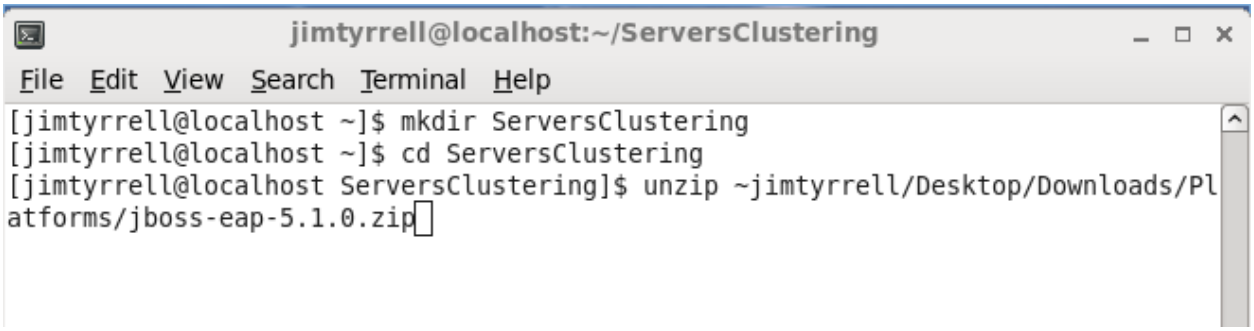
```
jboss-eap-5.1.0.zip
```

Just Unzip and Go

Installing the EAP is very very simple, and has the following high level steps:
Create a ServersClustering directory in the user home directory, make this unique
Unzip the contents of the file above into that directory

mkdir ~student/ServersClustering
```
cd ~student/ServersClustering
unzip ~student/Desktop/Downloads/Platforms/jboss-eap-5.1.0.zip
```

Your command/s should look something like this:



Make sure you hit enter after the unzip command and wait for it to finish.

That is it, now JBoss Enterprise Application Platform is installed and ready to use. You should feel pretty good that in less then a few minutes you have installed a full JEE container.

Apache Portable Runtime

The next step is to install the APR or Apache Portable Runtime into the Container. This is not specifically needed for setting up clustering, but performance and getting the most out of your available resources is something customers typically need/want when setting up clustering. In order to install the APR, you just need to unzip the included file from the Clustering folder.

The command to do this is:

```
unzip ~student/Desktop/Downloads/Platforms/jboss-ep-native-5.1.0-RHEL6-x86_64.zip
```

And it would look like this, the unzip is finished from the prior step, and you are now ready to unzip the Apache Portable Runtime Components:

```
  inflating: jboss-eap-5.1/seam/build/maven/README.txt
  inflating: jboss-eap-5.1/seam/build/maven/bin/mvn.bat
  inflating: jboss-eap-5.1/seam/build/maven/bin/mvnDebug.bat
  inflating: jboss-eap-5.1/seam/build/maven/bin/m2.conf
  inflating: jboss-eap-5.1/seam/build/maven/bin/m2.bat
[jimtyrrell@localhost ServersClustering]$ unzip ~jimtyrrell/Desktop/Downloads/Pl
atforms/jboss-ep-native-5.1.0-RHEL6-x86_64.zip
```

Now you have the first step for the install completed. You may ask yourself how do you know if you have the APR installed. Well at startup time the server will output a message letting you know, without the APR looks like this:

```
jimtyrrell@localhost:~/ServersClustering/jboss-eap-5.1/jboss-as/bin
File  Edit  View  Search  Terminal  Help
08:47:05,840 INFO  [AprLifecycleListener] The Apache Tomcat Native library which allows optimal
performance in production environments was not found on the java.library.path: /home/jimtyrrell/
JDKS/jdk1.6.0_23/jre/lib/amd64/server:/home/jimtyrrell/JDKS/jdk1.6.0_23/jre/lib/amd64:/home/jimt
yrrell/JDKS/jdk1.6.0_23/jre/../lib/amd64:/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/us
r/lib
08:47:05,886 INFO  [Http11Protocol] Initializing Coyote HTTP/1.1 on http-127.0.0.1-8080
08:47:05,886 INFO  [AjpProtocol] Initializing Coyote AJP/1.3 on ajp-127.0.0.1-8009
```

With the APR the output will look like this:

```
jimtyrrell@localhost:~/ServersClustering/jboss-eap-5.1/jboss-as/bin
File  Edit  View  Search  Terminal  Help
08:47:58,227 INFO  [AprLifecycleListener] Loaded Apache Tomcat Native library 1.1.19.
08:47:58,227 INFO  [AprLifecycleListener] APR capabilities: IPv6 [true], sendfile [true], random
 [true].
08:47:58,298 INFO  [Http11AprProtocol] Initializing Coyote HTTP/1.1 on http-127.0.0.1-8080
08:47:58,298 INFO  [AjpAprProtocol] Initializing Coyote AJP/1.3 on ajp-127.0.0.1-8009
```

Pretty obvious and at the end of your first server start you can scroll up to view the status of this. It should also be noted that the APR is specific to each Operating System, and you would have to get the correct one.

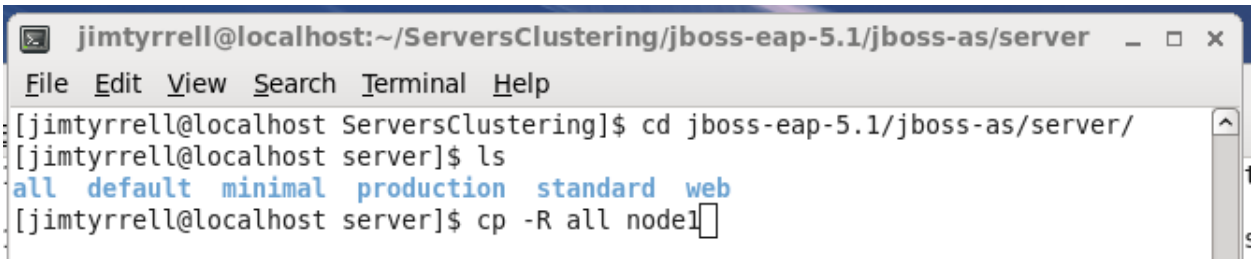
That is it for unzipping files and having all the files you will need on your local file system as required for the next steps in the lab. If you are confused or something does not feel right, please feel free to raise your hand.

Copy a Server Config

All changes we plan to make to our server configuration will be a derivative of the all configuration that ships with JBoss. If you are not familiar JBoss makes it very easy to create specialized configurations, and ships with several out of the box including: all, production, default, etc. We will simply change to the correct directory and execute a copy.

cd ~student/ServersClustering/jboss-eap-5.1/jboss-as/server/
`cp -R all node1`

As shown below:



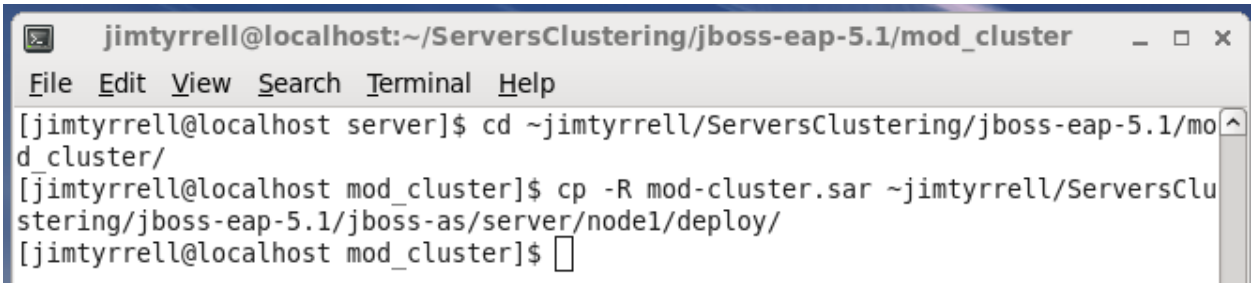
Configuration File Changes

JBoss is just a simple set of files that can be changed. Once these changes are made you can zip up or copy that configuration and make it available as golden image. We will use that feature a little later on.

First we need to copy the mod_cluster.sar file which was delivered in the main jboss-eap-5.1.0.zip in the mod_cluster directory:

cd ~student/ServerClustering/jboss-eap-5.0/mod_cluster
`cp -R mod-cluster.sar ~student/ServersClustering/jboss-eap-5.1/jboss-as/server/node1/deploy`

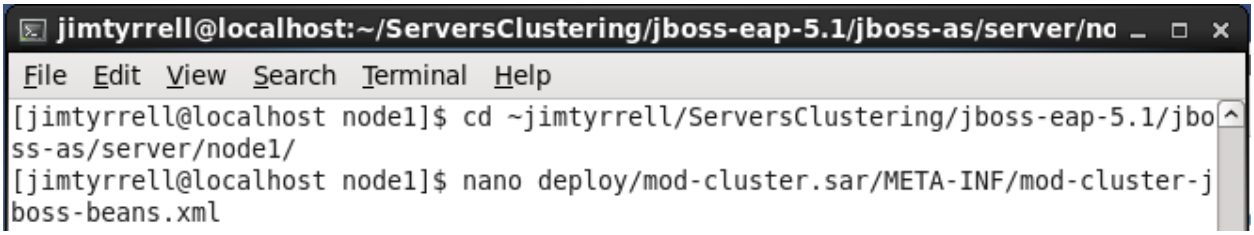
As shown:



Next we need to edit the mod-cluster-jboss-beans.xml file, you can use your favorite editor for this vi or nano. If you know vi your all set, if not nano is right up your alley.

```
cd ~student/ServerClustering/jboss-eap-5.0/jboss-as/server/node1
nano deploy/mod-cluster.sar/META-INF/mod-cluster-jboss-beans.xml
```

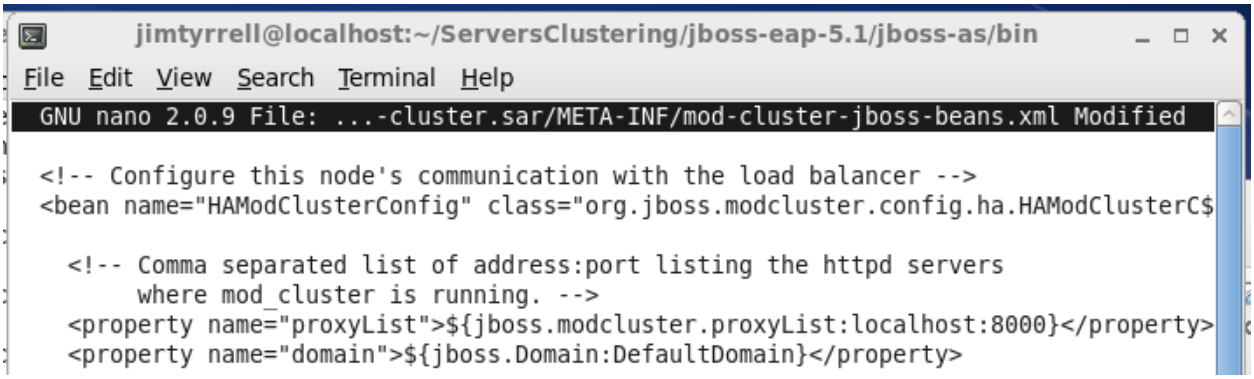
As shown:



Next you need to edit and/or add two lines in the file, scroll down until you find the entry for ..."proxyList"..
Those two lines will look like this:

```
<property name="proxyList">${jboss.modcluster.proxyList:localhost:8000}</property>
<property name="domain">${jboss.Domain:DefaultDomain}</property>
```

So it ends up looking like this:

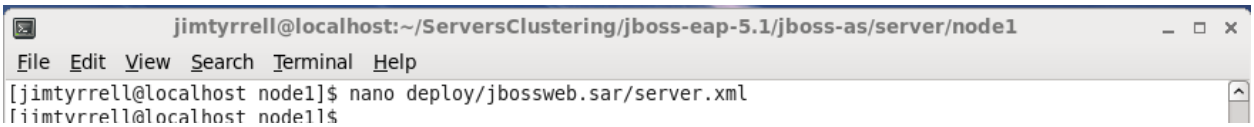


Make sure you save the file.

Next you will have to edit the server.xml file in the jbossweb.sar file, that command will look something like this, again use which ever editor you are most comfortable with.

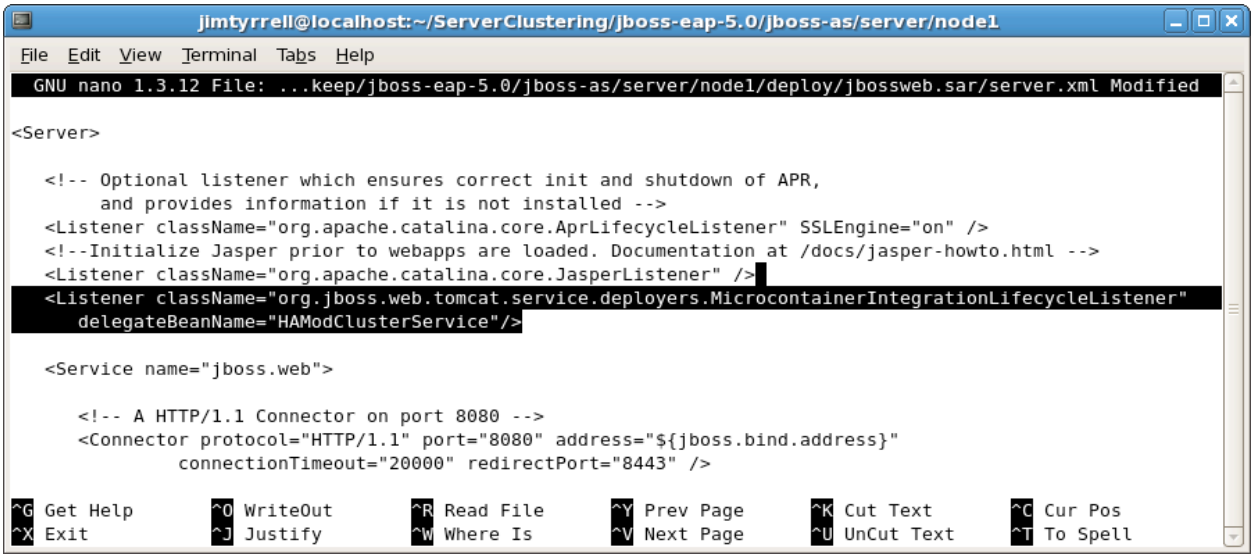
```
nano deploy/jbossweb.sar/server.xml
```

It will look something like this:

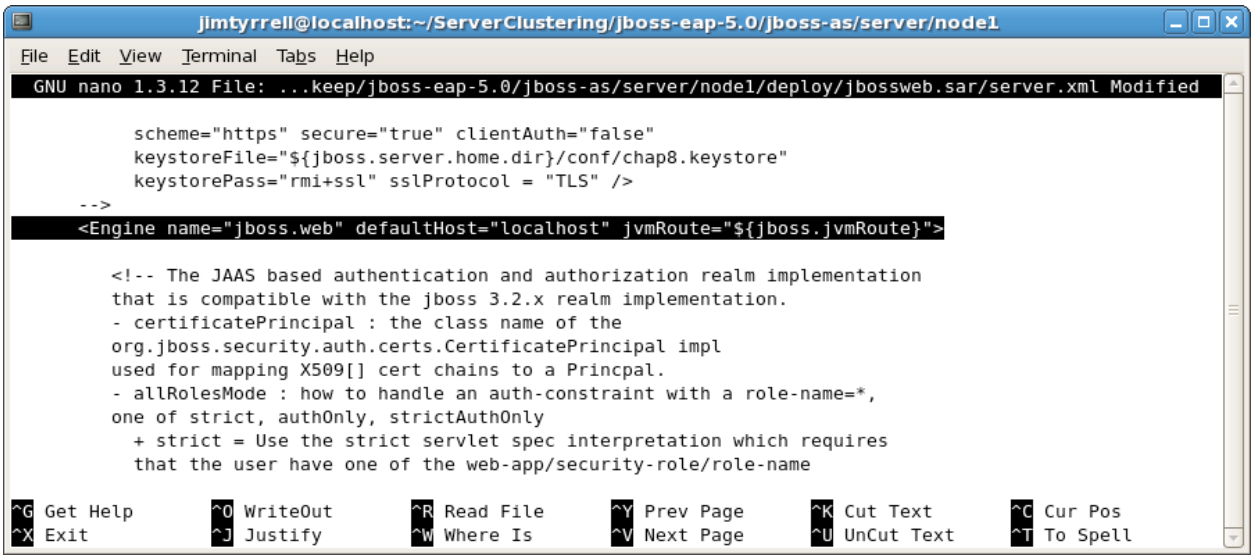


You then need to add/edit two lines in this file, the first is to add a new Listener near the other listeners at the top of the file and the second is to add a jvmRoute to the existing Engine Component

```
<Listener
className="org.jboss.web.tomcat.service.deployers.MicrocontainerIntegrationLifecycleLi
stener" delegateBeanName="HAModClusterService"/>
```



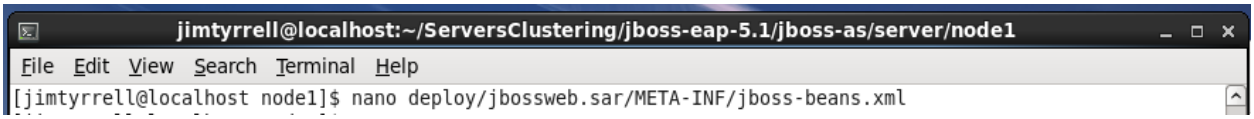
```
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="${jboss.jvmRoute}">
```



After making those two changes make sure you save out the file.

The next step is to edit the jboss-beans.xml file

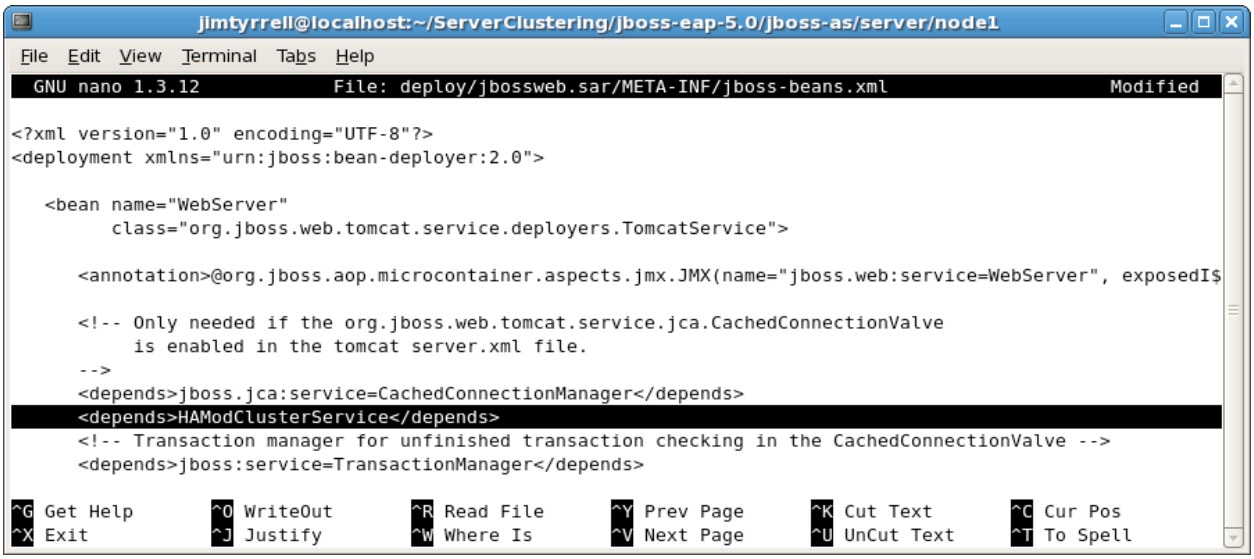
```
nano deploy/jbossweb.sar/META-INF/jboss-beans.xml
```



and add in this entry:

```
<depends>HAModClusterService</depends>
```

It should look like this:



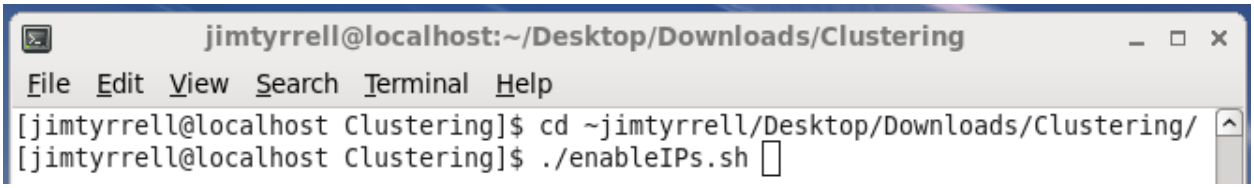
Make sure you save the file.

Moment of Truth

Now it is time to see if all of the changes you made were successful. In other words can you start up the jboss instance. Before you do that you need to create a few ip address and multicast address on your local box. You can open up the readme.txt in the ~student/Desktop/Downloads/Clustering/ directory and you will find several commands you need to run to turn on these ip address. Or even easier you can just run the command:

```
./enableIPs.sh
```

as shown:



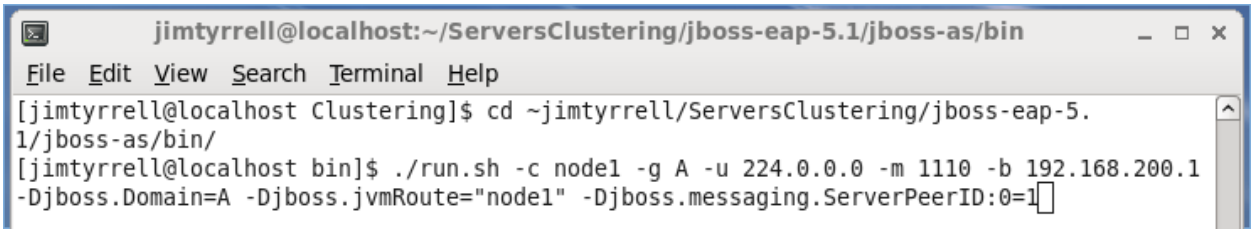
It will ask you for your password as you should be in a lab added to the sudoers file. If you are on your own system you will have to ask your administrator how to turn on these IP Address.

The next step is to change to the bin directory and start up node1. The readme has lines for starting each of the four nodes we will start eventually. Grab the first one and lets start up the server:

```
cd ~student/ServerClustering/jboss-eap-5.0/jboss-as/bin
./run.sh -c node1 -g A -u 224.0.0.0 -m 1110 -b 192.168.200.1 -Djboss.Domain=A
-Djboss.jvmRoute="node1" -Djboss.messaging.ServerPeerID:0=1
```

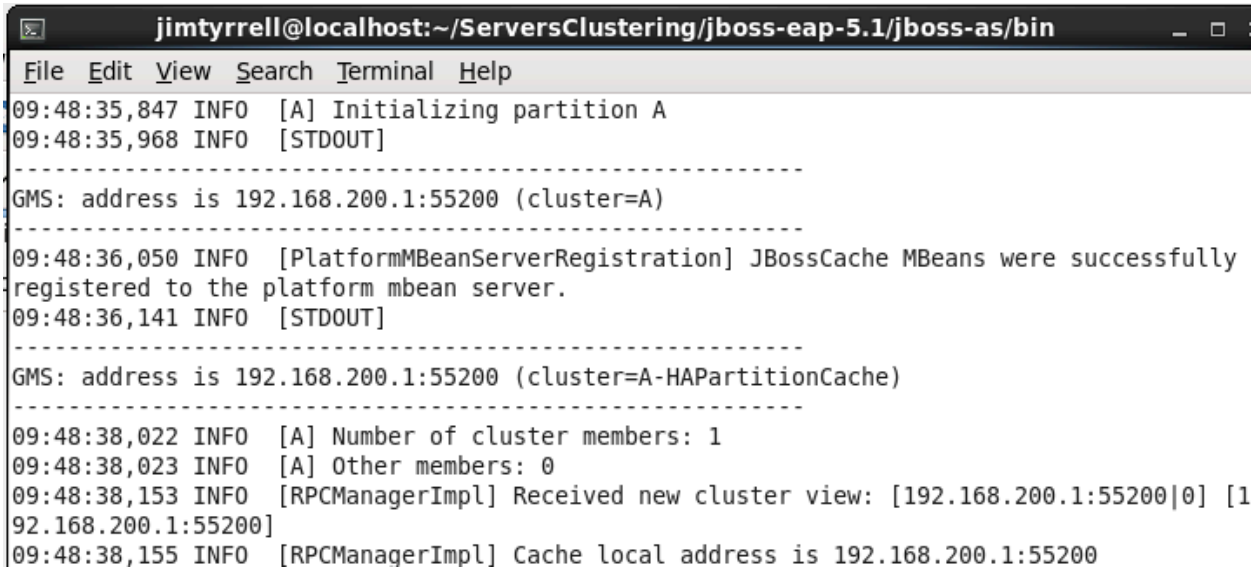
```
You may be wondering about all those options above, lets break them down:
-c is for configuration
-g is for group name in clustering
-u is for unicast address
-m is for multicast port address
-b is for IP Address
-Djboss.Domain is a unique domain used in mod_cluster
-Djboss.jvmRoute is used to uniquely identify a worker node
-Djboss.messagin.ServerPeerID:0 is used to inject a unique number into JBoss Messaging
```

As shown:

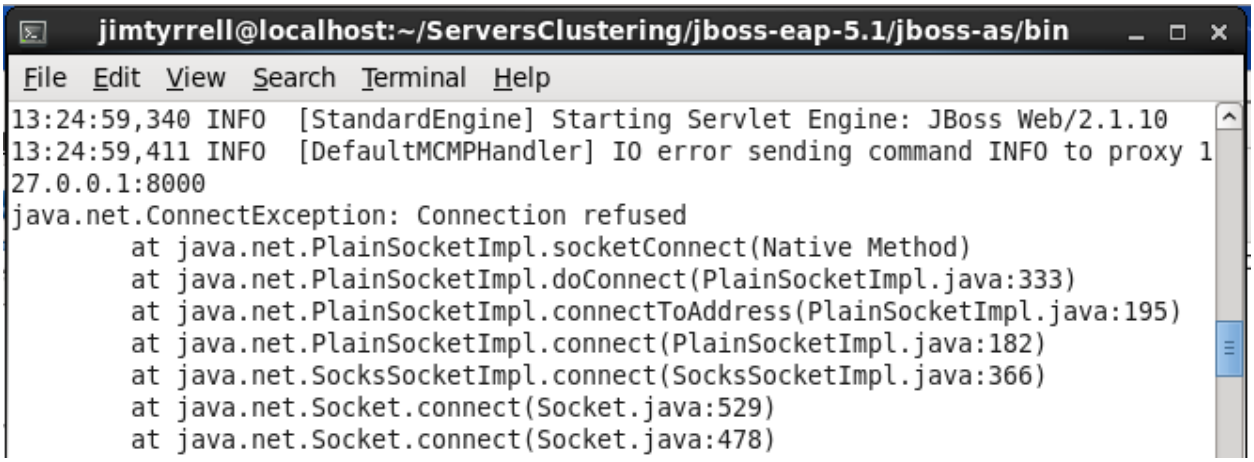


Hit enter and wait for the server to come up. Remember to scroll up and see that the APR was installed, was it?

A few things to note, you should see that this server is a member of a cluster of one as shown:



You will see an error like this, it is okay as we have not yet setup apache:

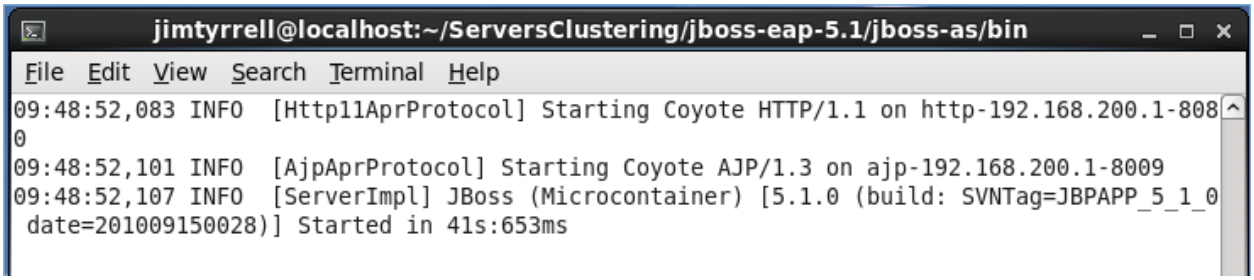


The above error is okay, any other errors are not acceptable, if you have any other errors please raise your hand.

When you see this message the server has finished starting:

```
......Started in .....
```

As shown:



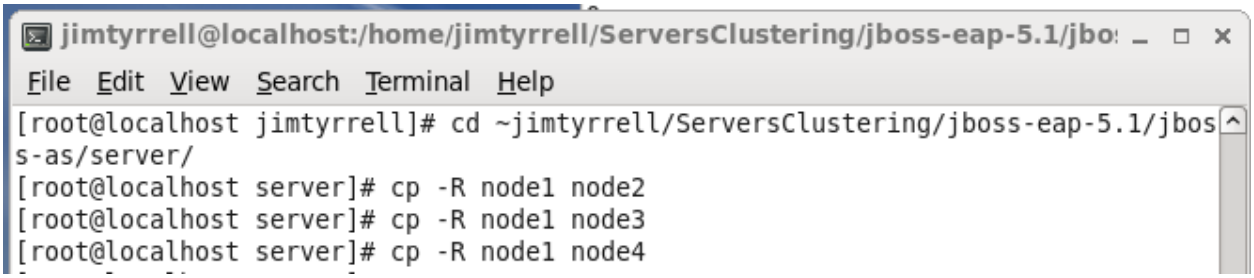
You have now started node1, congratulations.If you remember how you copied “all” into “node1” earlier, if you have this server running correctly, it is time to do that to create node2, 3, and 4.

Simple cd to the server directory and execute:

cd ~student/ServersClustering/jboss-eap-5.1/jboss-as/server/

```
cp -R node1 node2
cp -R node1 node3
cp -R node1 node4
```

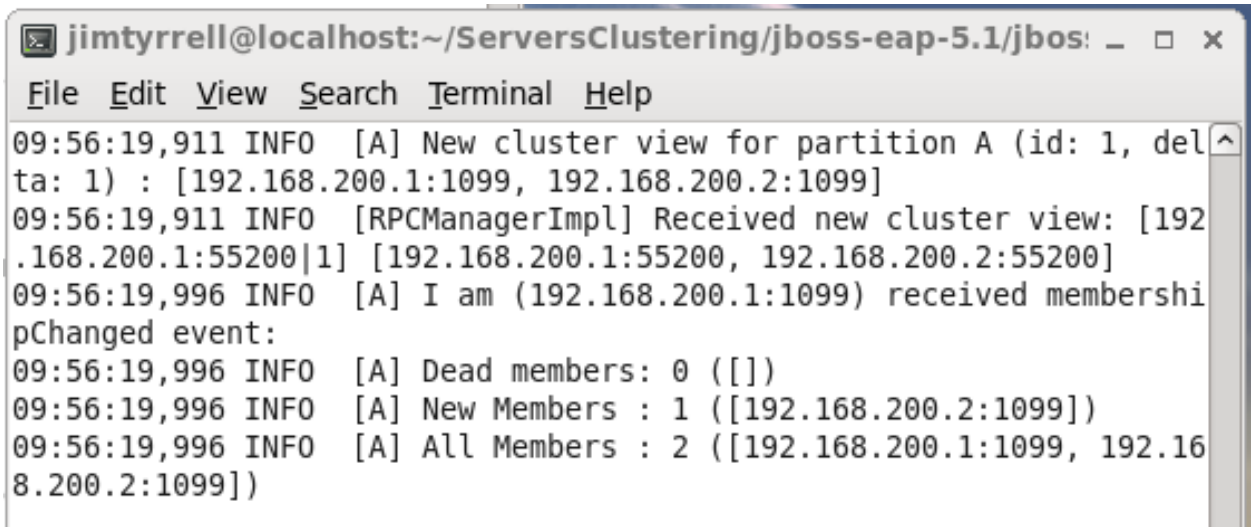
It should look like this when you are done:



Start up node2 using the second startup command from the readme file:

```
cd ~student/ServerClustering/jboss-eap-5.0/jboss-as/bin
./run.sh -c node2 -g A -u 224.0.0.0 -m 1110 -b 192.168.200.2 -Djboss.Domain=A
-Djboss.jvmRoute="node2" -Djboss.messaging.ServerPeerID:0=2
```

Make sure it starts without any errors, other then the one noted above. Also note that it joined a cluster, with messages in both windows that looked like this:
Your first server you started will look like this:



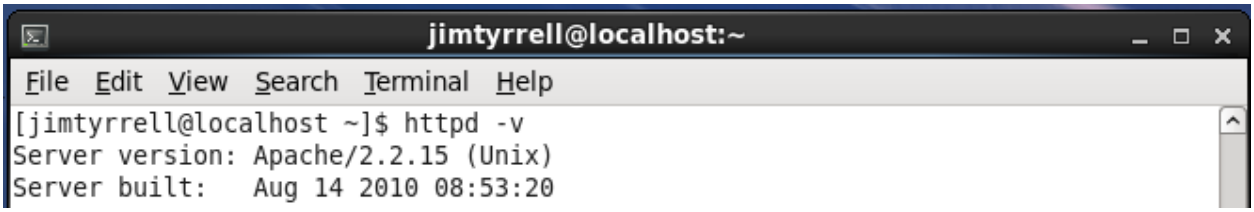
As we shut down and play with these instances these messages will let you know when a machine has left the cluster.

Congratulations you now have two servers that are clustered together, but you do not have any load balancing for web content going on between them. That we will cover in the next lab.

Lab Number 2: Run/Install Apache HTTPD

Apache HTTPD should already be installed and available on your RHEL 6 instance if this is a Red Hat run lab. If not make sure you have either RHEL 6 with Apache HTTPD installed, a supported Enterprise Web Server for a fully supported configuration for other operating systems, or at miniunum a Apache HTTPD 2.2.8 or higher installed. You will also need the appropriate native components downloaded and available, in this lab we have already done that in Lab Number 1.

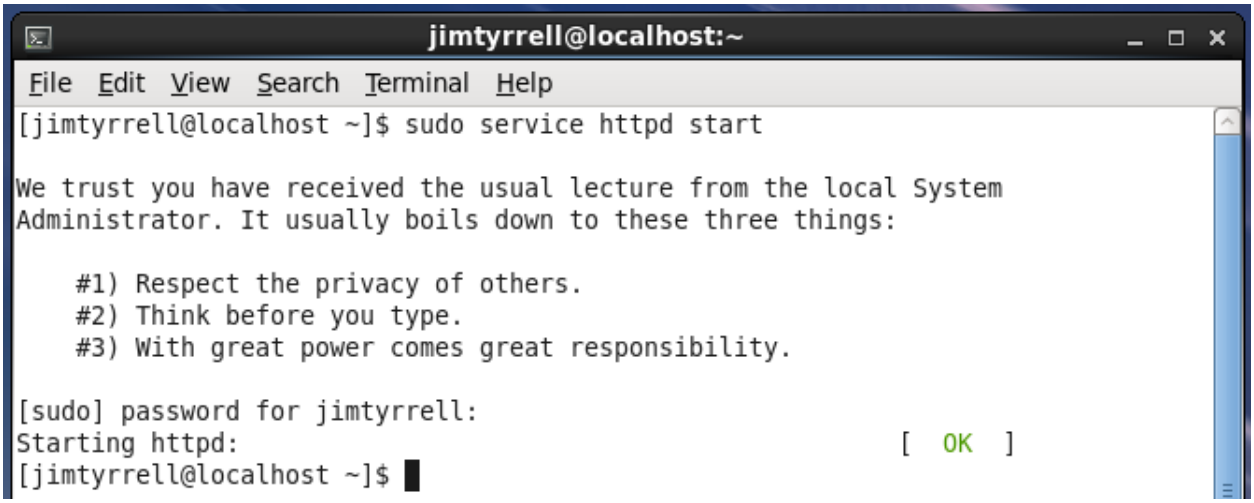
At the command line to view the currently installed httpd server, you can run httpd -v as shown:



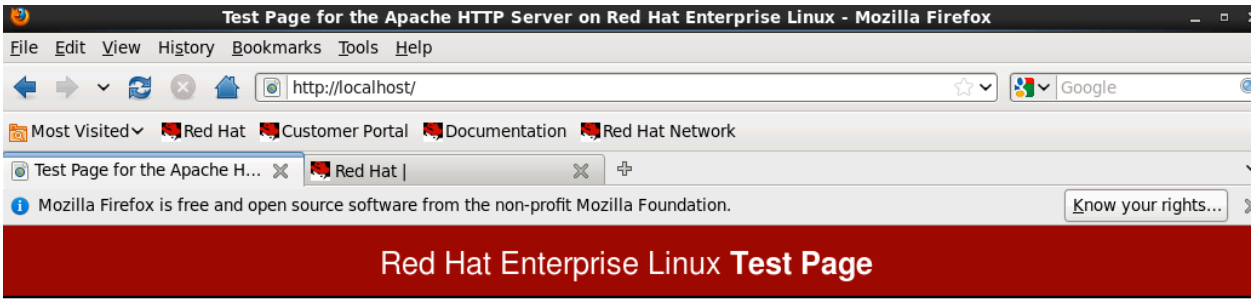
If you open a web browser on an instance of Red Hat Enterprise Linux 6 without doing anything, you will probably find that the httpd server has not been stared. To start it run:

```
sudo service httpd start
```

As shown:



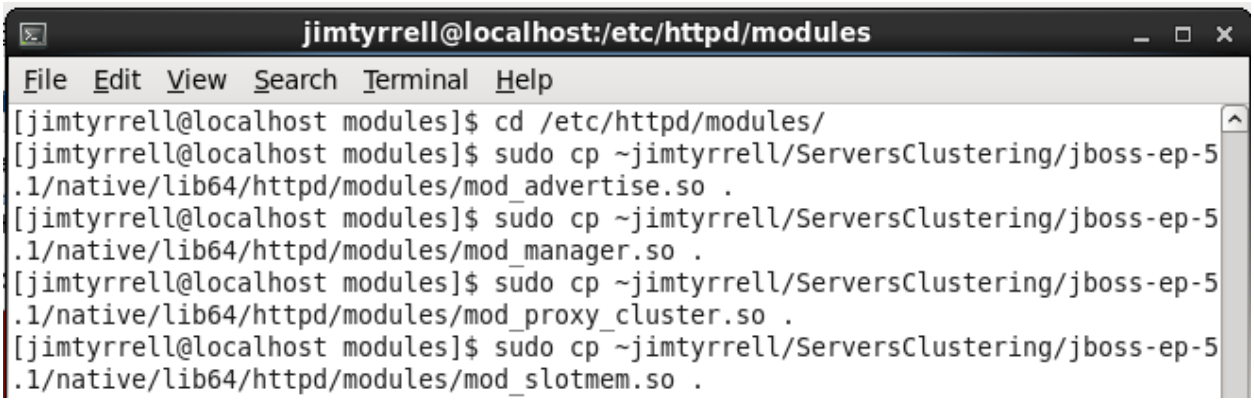
You should open a web browser and make sure that you can see the web server via http://localhost as shown:



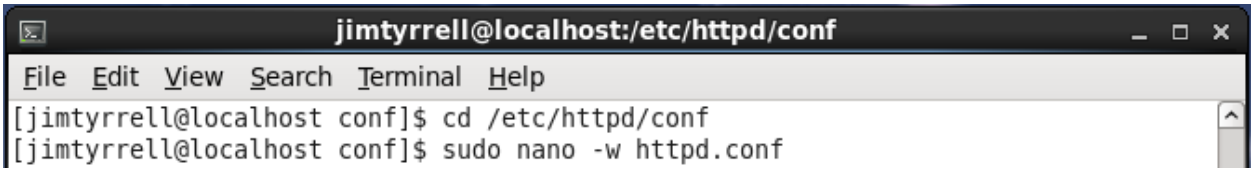
Now it is time to install mod_cluster so it can act as a proxy sending out requests to the installed/discovered JBoss Enterprise Application Platform Instances.

The first step is to copy the four native files that are needed into the apache modules folder.

```
cd /etc/httpd/modules/
sudo cp ~student/ServersClustering/jboss-ep-5.1/native/lib64/httpd/modules/mod_advertise.so .
sudo cp ~student/ServersClustering/jboss-ep-5.1/native/lib64/httpd/modules/mod_manager.so .
sudo cp ~student/ServersClustering/jboss-ep-5.1/native/lib64/httpd/modules/mod_proxy_cluster.so .
sudo cp ~student/ServersClustering/jboss-ep-5.1/native/lib64/httpd/modules/mod_slotmem.so .
```

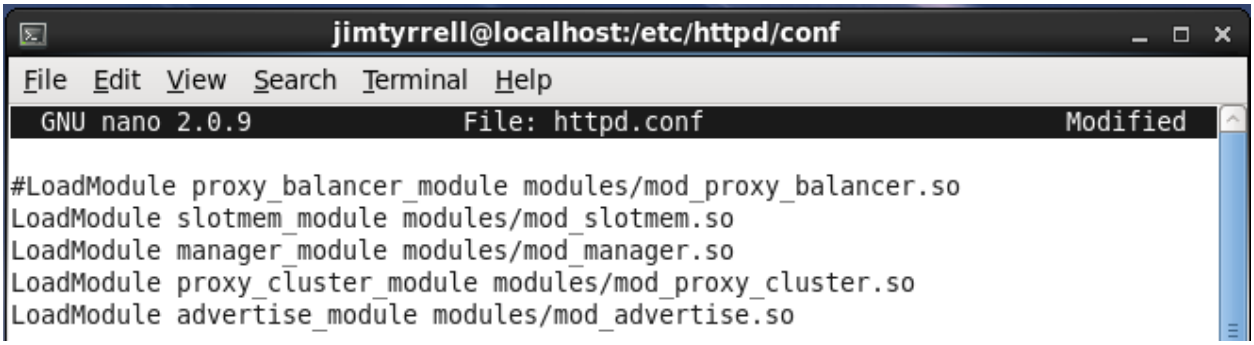


Next edit the httpd.cnf file as shown, make sure you use sudo to edit the file as you can see below:



Comment out an existing line, and add in four new lines:

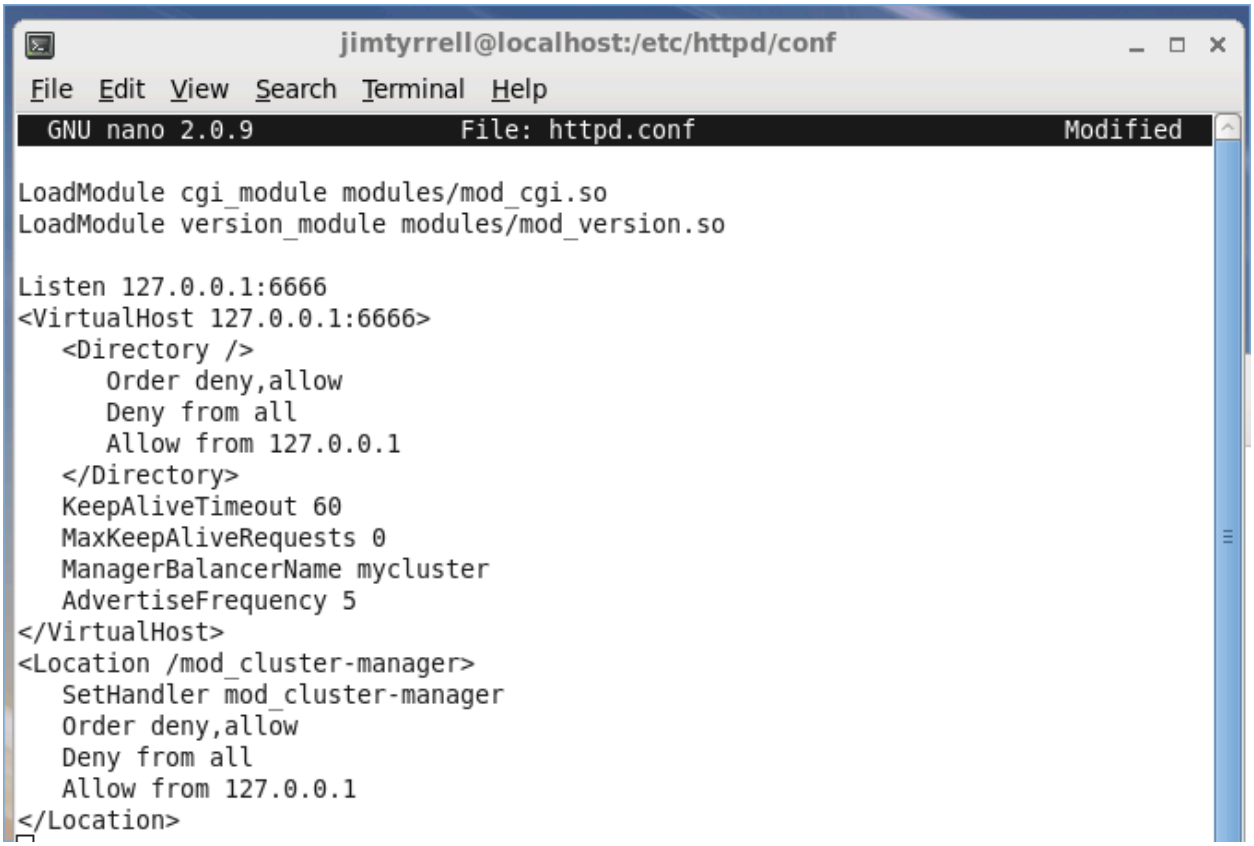
```
#LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so
```



In the same file add in the below section as shown:

```
Listen 127.0.0.1:8000
<VirtualHost 127.0.0.1:8000>
   <Directory />
       Order deny,allow
       Deny from all
       Allow from 127.0.0.1
   </Directory>
   KeepAliveTimeout 60
   MaxKeepAliveRequests 0
   ManagerBalancerName mycluster
   AdvertiseFrequency 5
</VirtualHost>
<Location /mod_cluster-manager>
   SetHandler mod_cluster-manager
   Order deny,allow
   Deny from all
   Allow from 127.0.0.1
</Location>
```

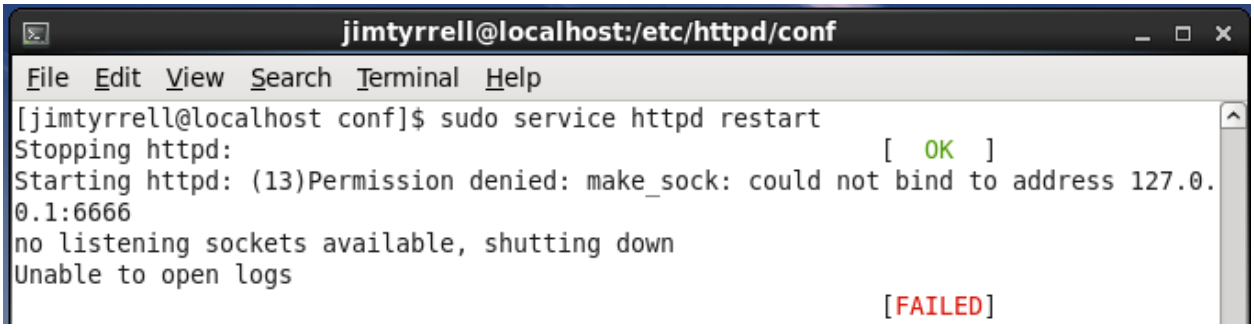
At the end of the LoadModule Section you can add the above as shown:



Now you should run:

```
sudo service httpd restart
```

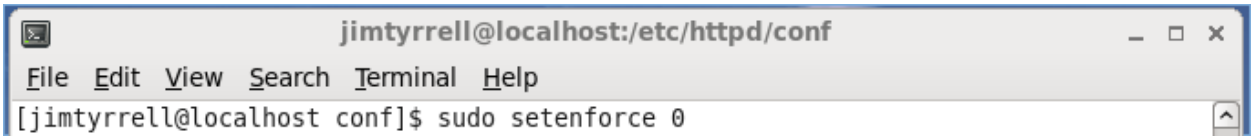
You will note that you will get an error, as by default SE Linux is enabled in Red Hat Enterprise Linux 6. For the purposes of this lab, we are going to turn this off, however, in a real production setting you would not want to do this. Appendix A will walk you through how to enable SE Linux. There are many reasons for using SE Linux that are far outside the scope of this Lab. The error you will see looks something like this:



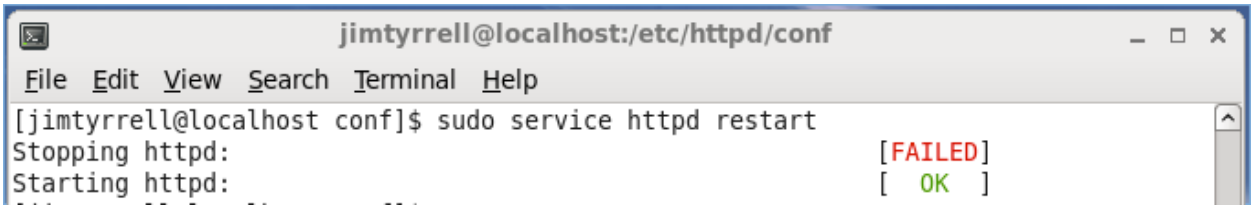
In order to fix this turn off SE Linux by running the command:

```
sudo setenforce 0
```

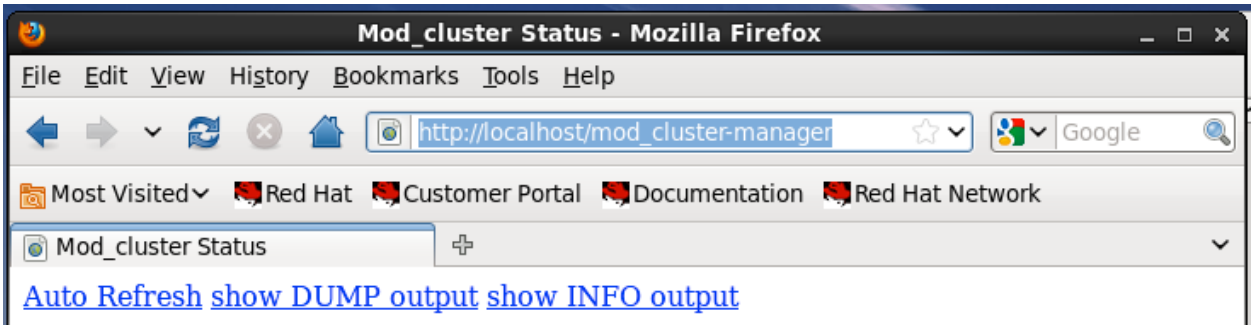
as shown:



Now rerun the sudo service httpd restart command and you should have success as shown:



Now check out the mod_cluster console open http://localhost/mod_cluster-manager as shown:



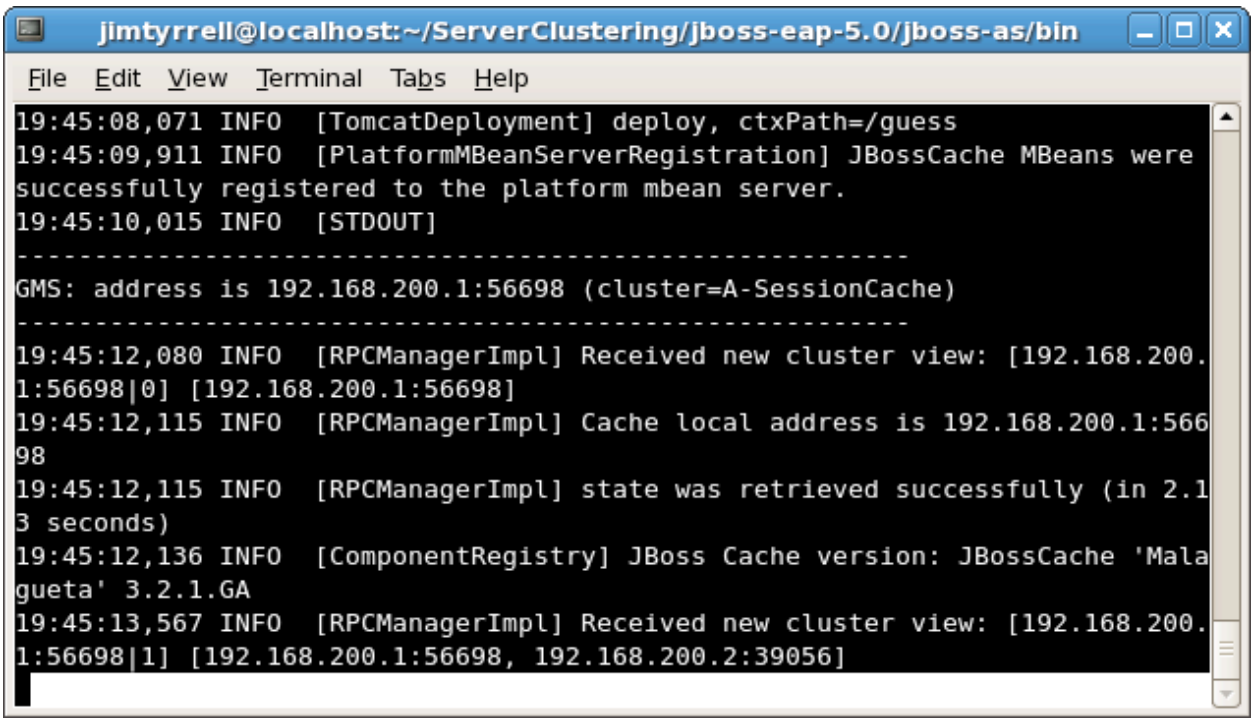
Lab Number 3: Deploy your First Application

Caution: For the ease of the lab we will be using the farm directory to replicate war files for us, however under no c~student/Desktop/Downloads/Clusteringircumstances should you ever use that method in production.~student/Desktop/Downloads/Clustering

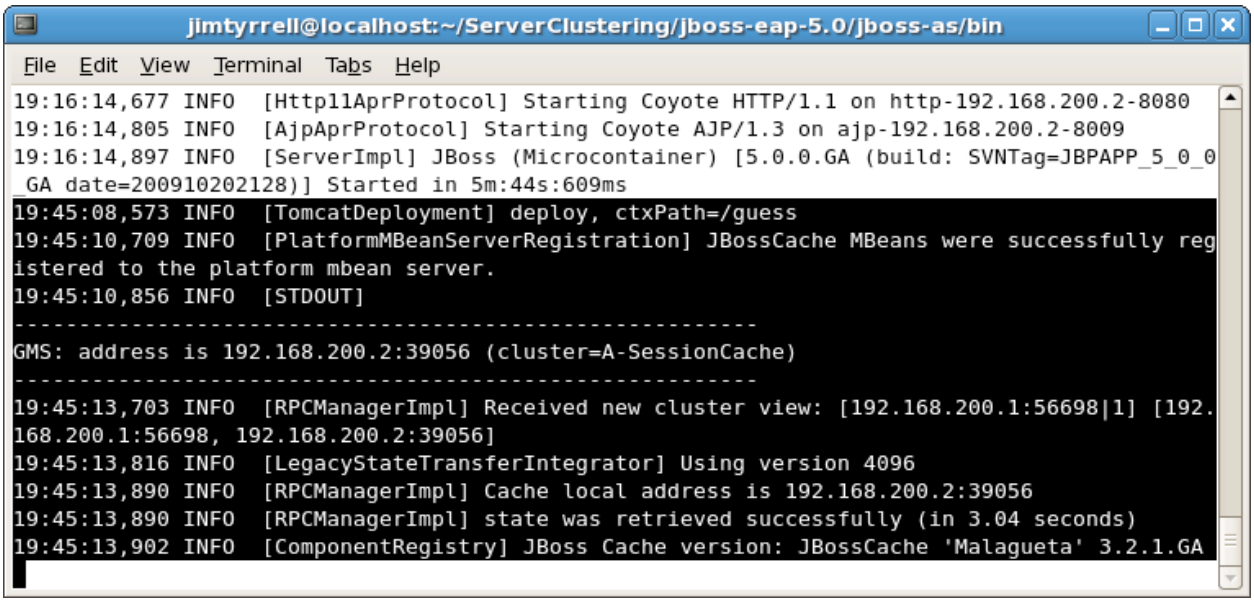
In your ~student/Desktop/Downloads/Clustering/session-demo-wars folder are there war files guessv1.war, guessv2.war and guessv3.war. You just need to copy the guessv1.war file into the farm directory of node1.

```
cp -R guessv1.war
~student/ServersClustering/jboss-eap-5.1/jboss-as/server/node1/farm/guess.war
```

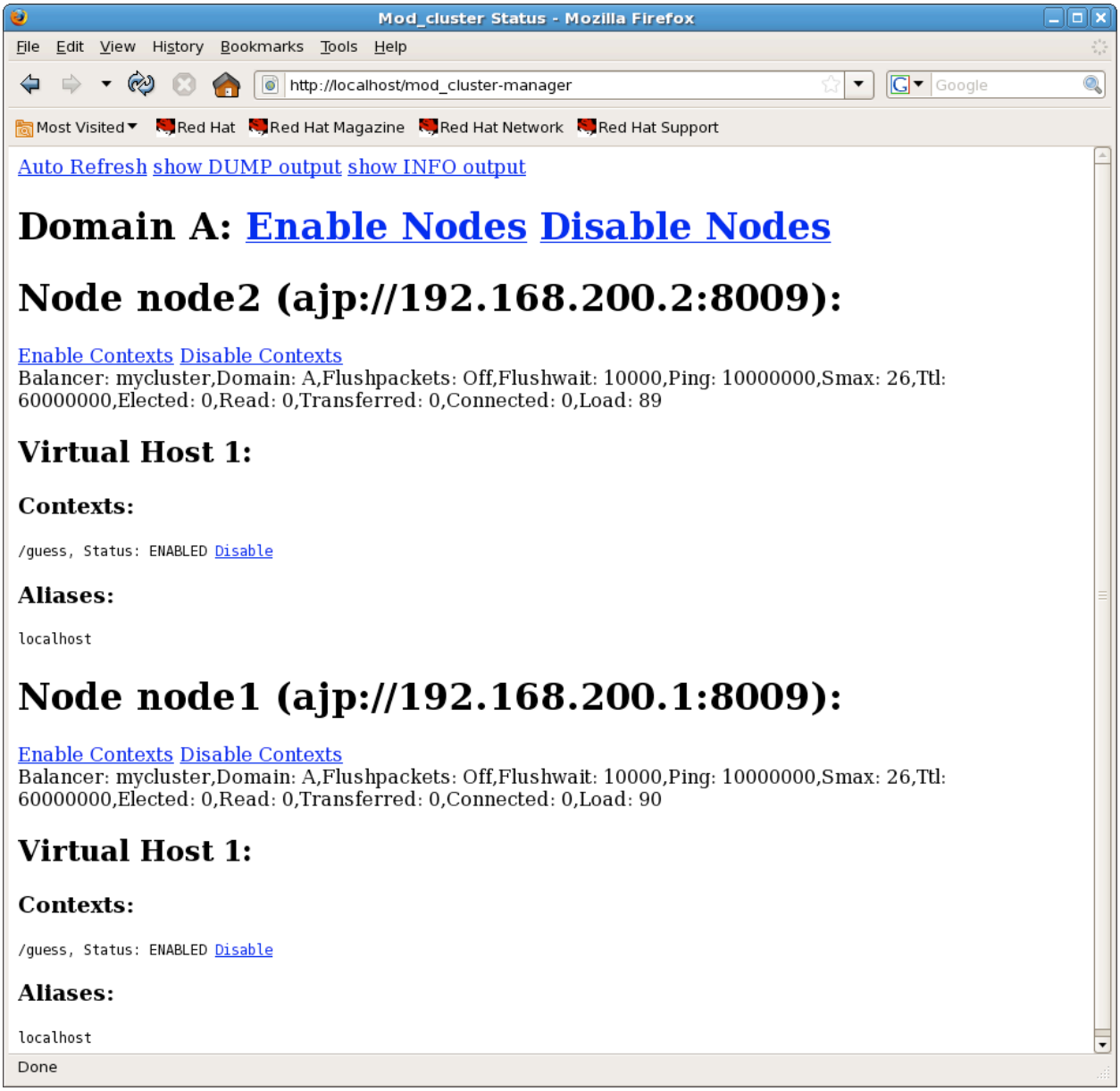
You can see that node1 deployed the application and also copied it via the farm service to node2 in the next screenshot:



You can see that node2 received the file and started up the application:



Now refresh your browser window, and you should see the two nodes each with a new /guess context as shown below:

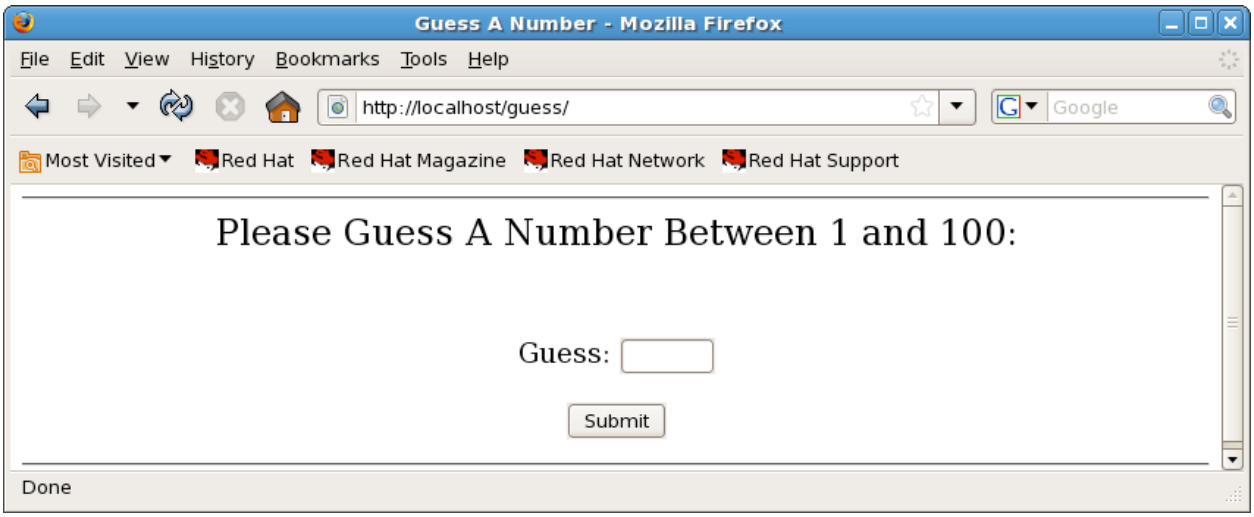


Congratulations you now have the applications deployed and you are now ready to see the load balancing in action. If your web browser does not look like this, please raise your hand.

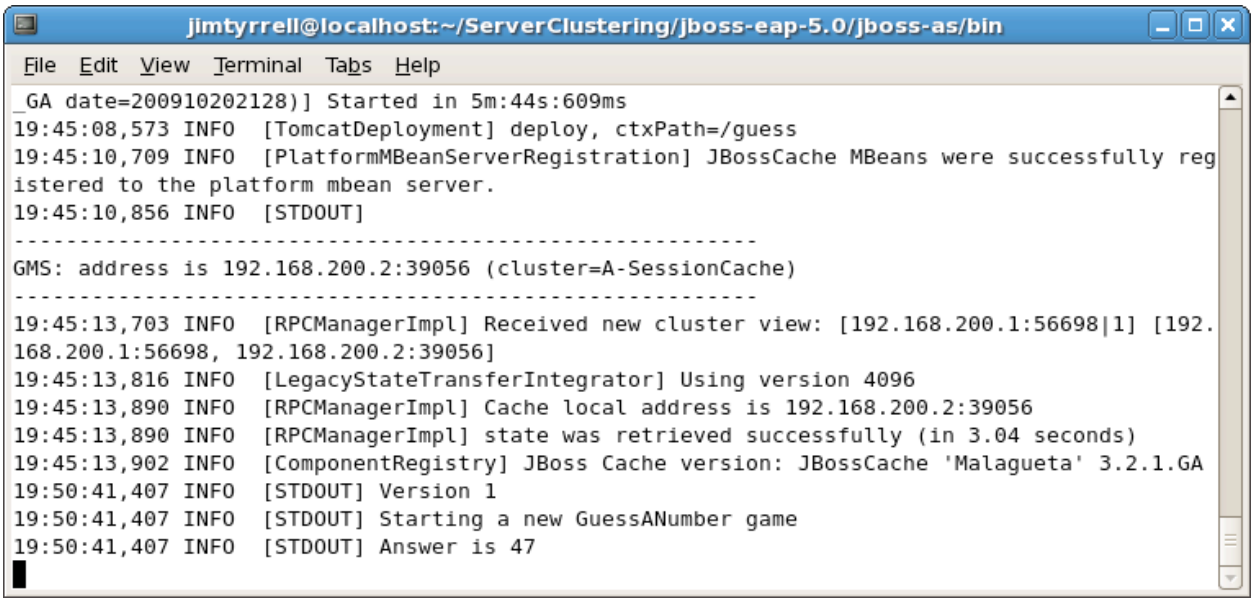
Lab Number 5: Bounce and Play

Test Load Balancing

Load up the URL http://localhost/guess as shown below:

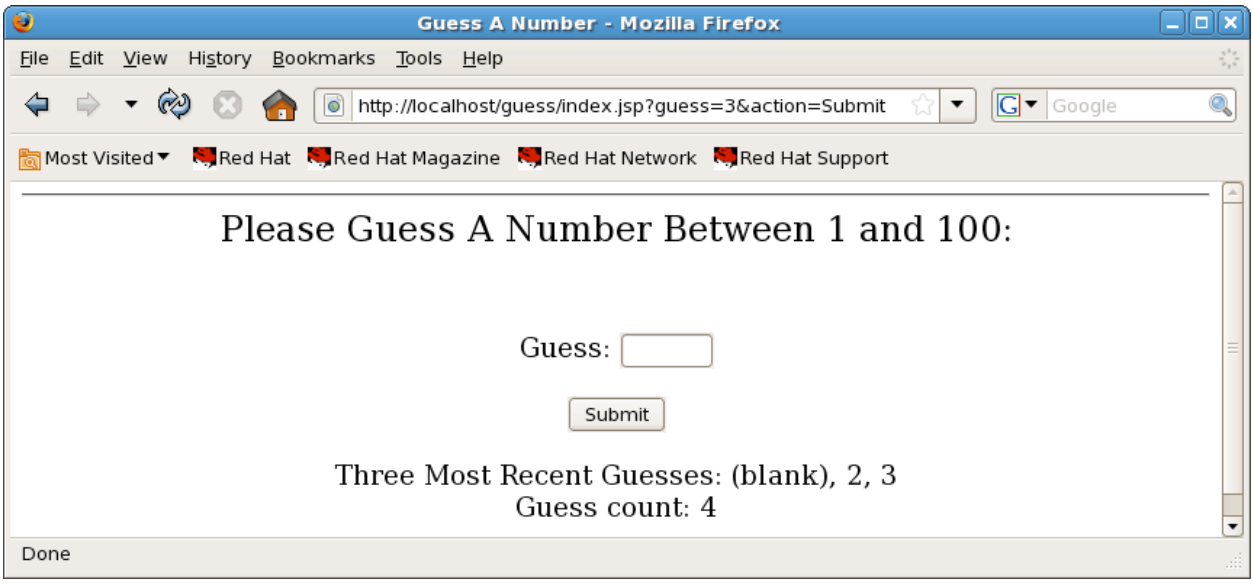


Also note that one of your node terminals has some output that looks like this:

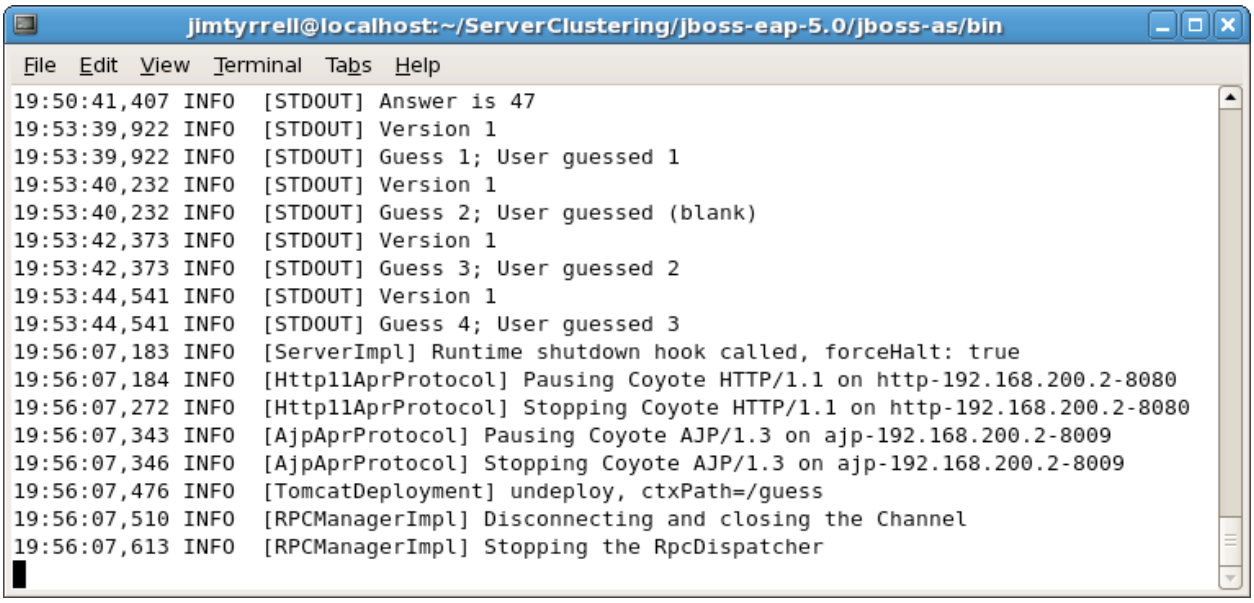


This is the starting of a new session that is unique to that server, however under the covers this stats has been replicated. Make at least three guesses in the web browser, note the answer is above, not the most secure application is it :). This is also the node that will serve the duration of a users requests until it is shutdown. This is done via sticky

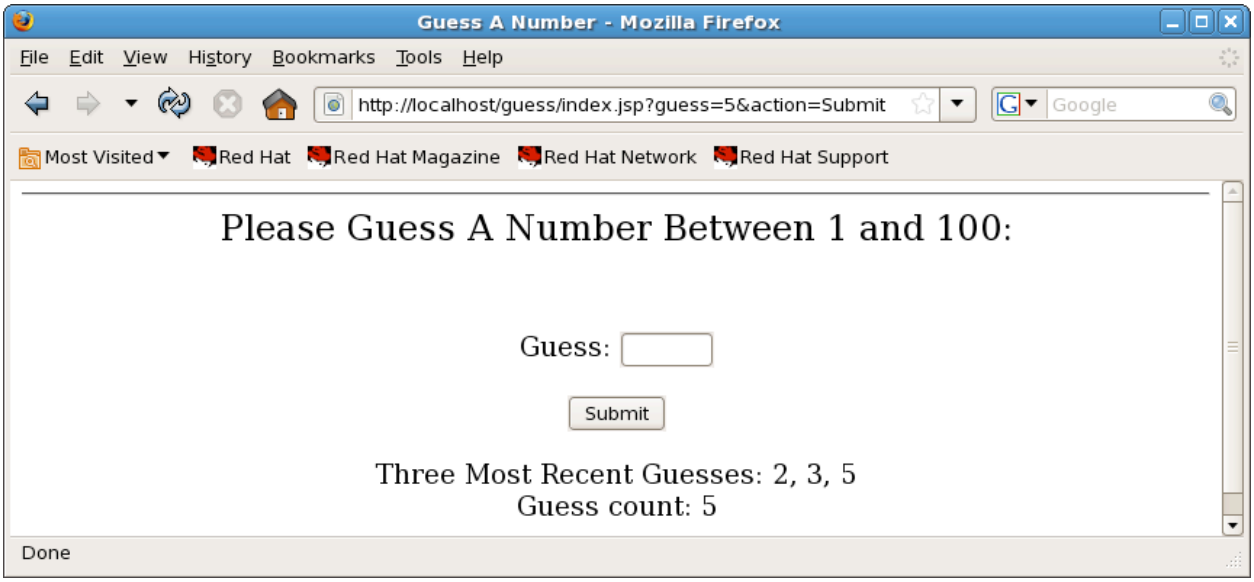
sessions. When your are all done your browser window will look something like this:



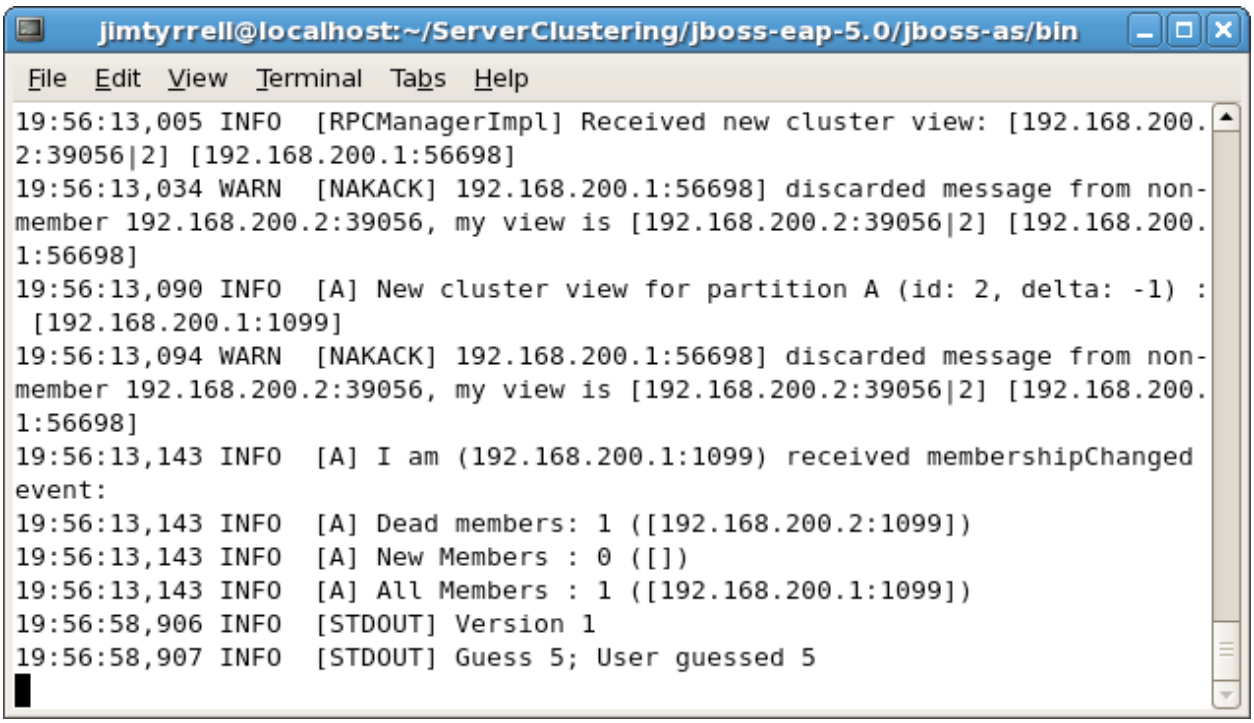
Now kill the node that was echoing the output shown above by press cntrl-c in the terminal window, it should look something like this:



You will now see some updated messages in the other node that was not hosting the work, submit a new guess in your browser window and see that the state was not lost even though you shutdown the server:



and the terminal showing the new guess:



Restart by up arrowing to the server start command you just stopped with cntrl-c, wait until you see it has joined the cluster and completely started, then kill the other server.

Now put in another guess and see how the load was automatically sent to the other server.

Extra Credit

A more advanced version of this lab, would be to start up node3 with the third ./run.sh file in the readme.txt and see how it dynamically joins the cluster. When it is fully started feel free to kill the other two nodes.

The steps are outlined below:

1. Start up the third server, notice how the guess.war file was copied for you.
2. Refresh the `http://localhost/mod_cluster_manager` to see how the third server has joined the cluster.
3. Kill the other one or two nodes once the third server has started (one or two depending on how you have been playing around with this)
4. Submit another guess and see how it has been load balanced

Congratulations you have now clustered and load balanced your war file across at least 2 servers, and possibly three if you did the more advanced part of the lab.

Feel free to play around with this some more.

When you are done make sure all three instances are closed down.

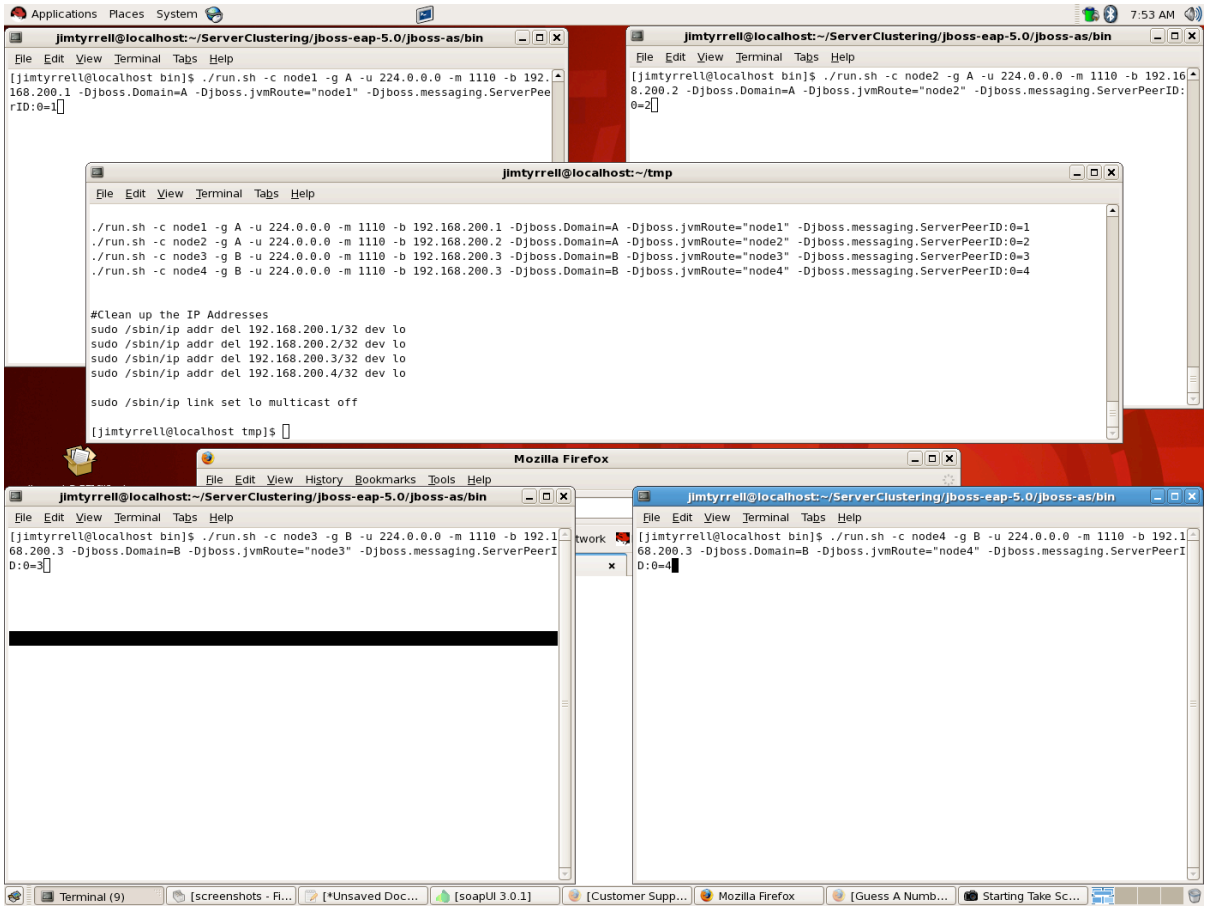
Lab Number 5: Rolling Server Restarts

Start Up the Servers

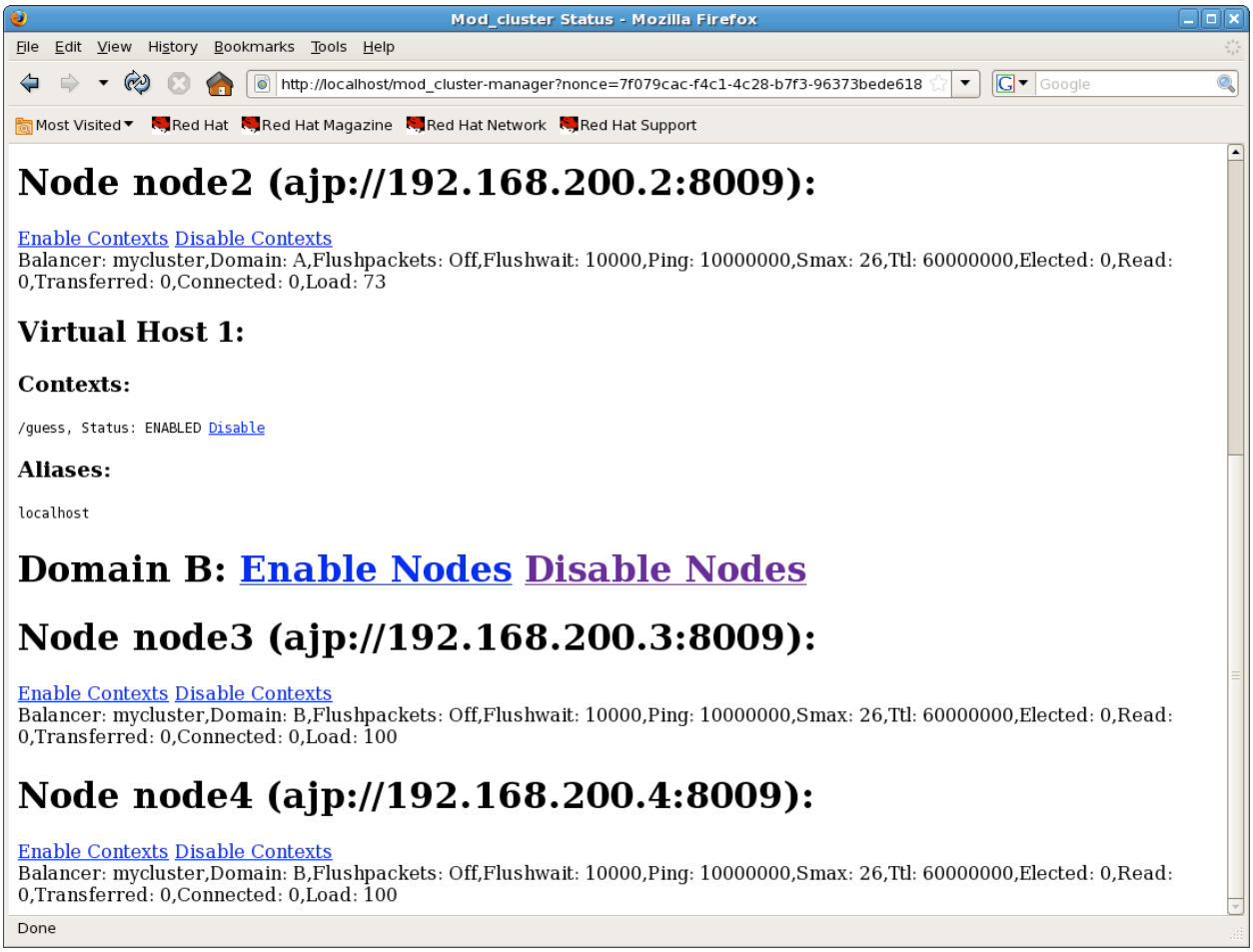
1. Open up the run.conf file in the jboss-as/bin directory and change the line JAVA_OPTS to have -Xms512m -Xmx512m instead of the defaults.

2. Open the readme.txt in the ~student/Desktop/Downloads/Clustering folder

3. Use the second set of server start commands to start four instances of JBoss, notice there is domain A and B int the startup commands. Also notice the different unique multicast addresses that are in use. Your screen should look something like this before starting all four terminals:



4. When you are done your http://localhost/mod_cluster-manger will have Domain A with the /guest application an, it will look about like this:

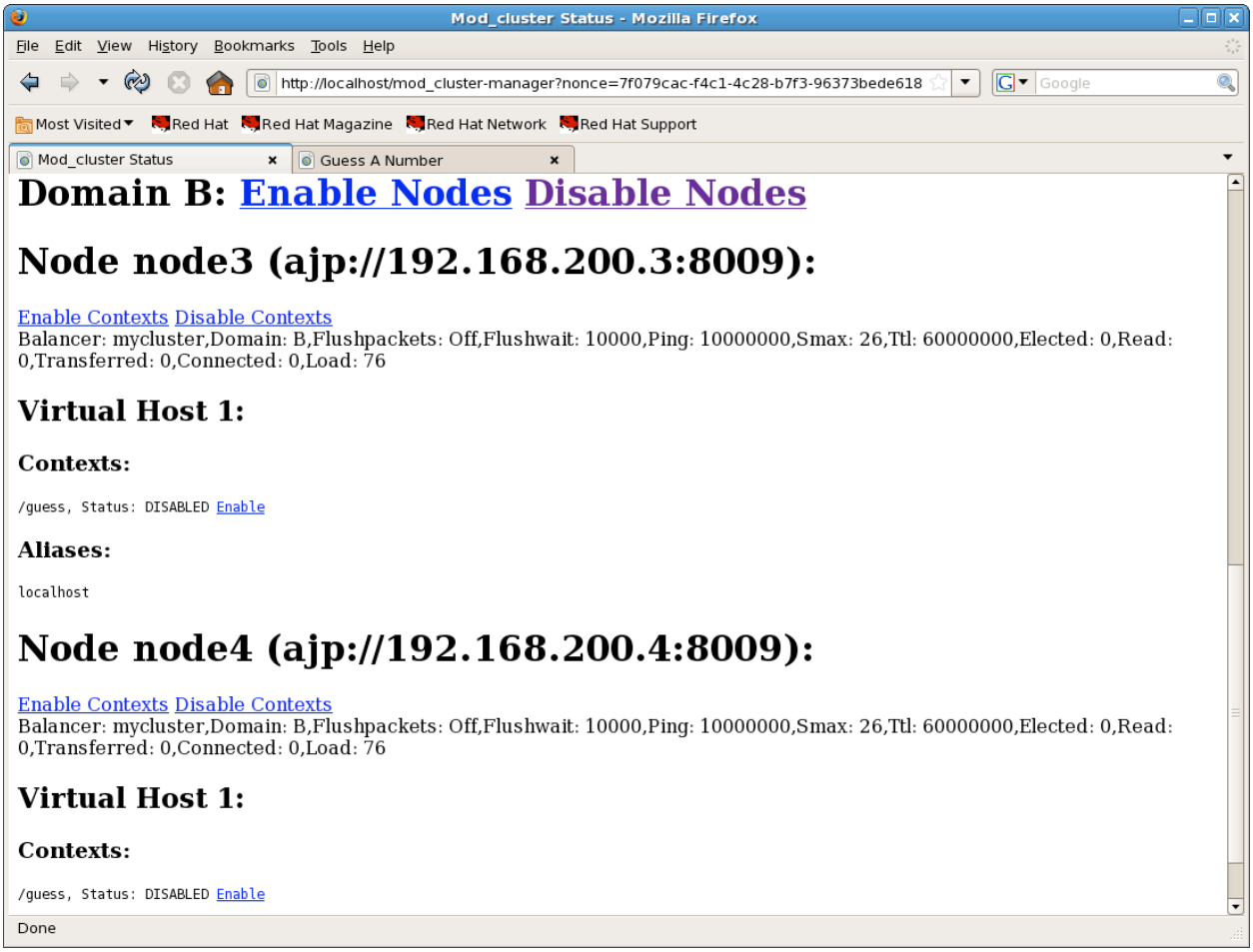


Now make some guesses as you did before, shut down the node that is hosting your application as before. Notice that it rolls over to the other node.

Now copy your guessv2.war file from the ${User_Home}/Downloads/Clustering/session-demo-wars into the node3/farm directory.

cp -R ${User_Home}/Downloads/Clustering/session-demo-wars/guessv2.war ~{User_Home)/ServerClustering/jboss-eap-5.0/jboss-as/server/node3/farm

Note that it was copied over to node 4 and refresh your mod_cluster-manager page, and click Disable nodes on Domain B. This will stop traffic from going to Domain B. It should look like this, check out the Disabled Contexts:



Now enable Domain B, by clicking Enable Nodes. Disable Domain A, by clicking Disable Nodes. Now go to your browser window and submit a new guess. Note the request was sent one of the Domain A Nodes as it was disabled and not stopped. Disabling the A node still allows it to service requests.

Now kill the remaining server in Domain A. The session will failover to Domain B, but you will probably see an exception. Did you? This is because our application is not exactly coded to deal with this scenario. Doing live session migration is something that is generally pretty hard, and your applications need to be written to deal with this case.

Now continue on making guesses in Domain B, kill the server that is hosting the request, notice is fails over to the other node.

Now reverse the process, start up the two nodes in Domain A, leave the shutdown node in Domain B off.

Copy over version 3 of the guess.war file into the farm directory for node1 or 2 in Domain A.

Enable the Domain A in the mod_cluster-Manager, if it still shows disabled. Disable Domain B, and notice the failover and the fail.

You have now shown/demonstrated nearly 100% uptime with JBoss and rolling server migrations. You have now completed the clustering lab.

Conclusion

What you learned

- How to install JBoss for mod_cluster
- How to start Apache httpd
- Explore the mod_cluster-manager
- How to do live application migration and updating

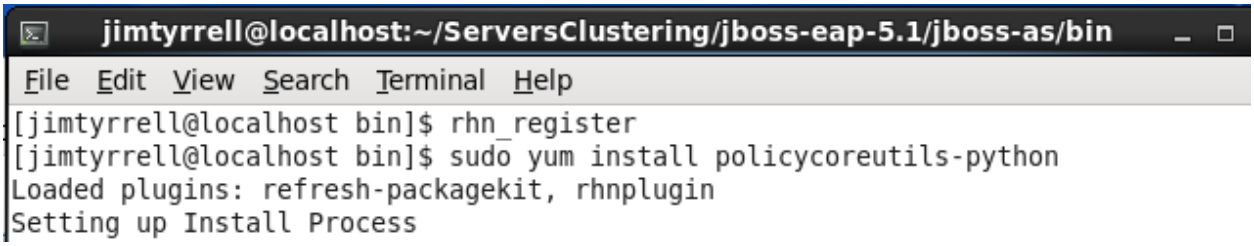
Appendix: SE Linux

Steps to use SE Linux
IMPORTANT READ THIS

Two bugs are filed against this, and check them out to see if this has been addressed before going on with these next steps.
https://bugzilla.redhat.com/show_bug.cgi?id=675019
https://issues.jboss.org/browse/MODCLUSTER-211?focusedCommentId=12580036#comment-12580036

Earlier in this lab SE Linux was turned off to setup clustering for simple evaluation. For a production environment you will probably want to run SE Linux, the decisions and thoughts that go into that, are far outside the scope of this document. This document just addresses the how to enable, without any of the reasons as to why.

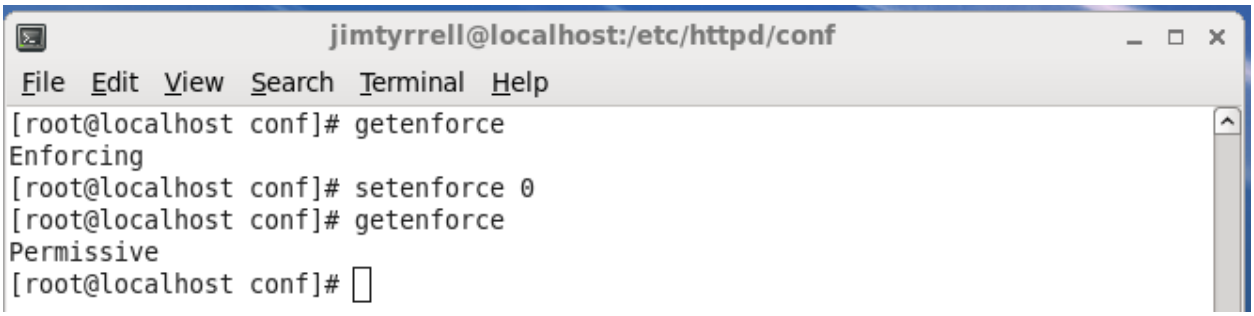
Install Some Python Utilities
`yum install policycoreutils-python, but first you might need to register your system via rhn_register as shown:`



First Turn Off SE Linux

Login as root.service
Earlier you turned off SE Linux, you can see its status by running the command, and to change it to Enforcing, run sudo setenforce 0 ie permissive mode, as shown:

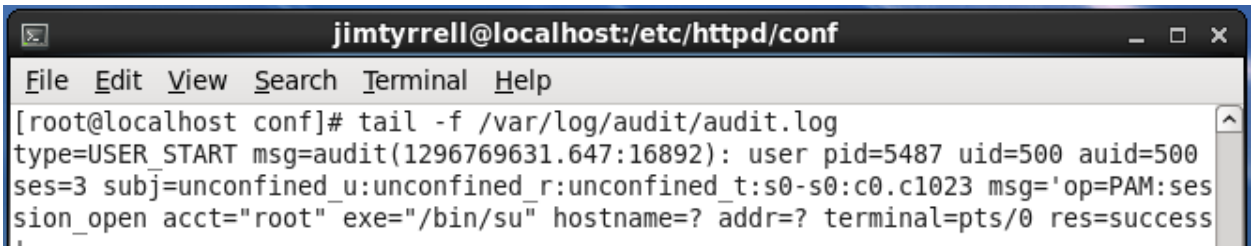
```
getenforce
setenforce 0
getenforce
```



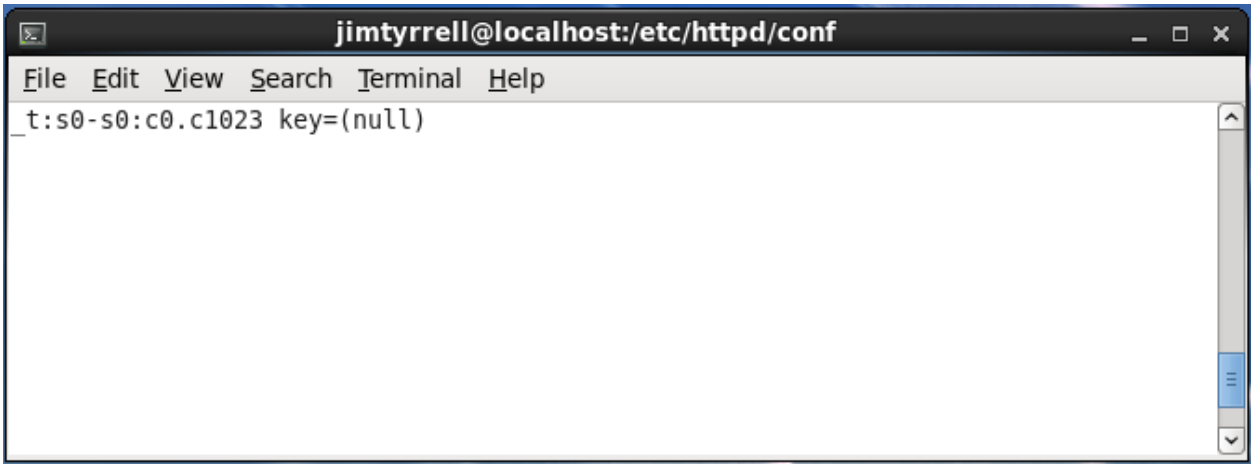
Create SE Linux Profile

First figure out how many audit messages are created when you try to start httpd with SE Linux turned off.
`tail -f /var/log/audit/audit.log`

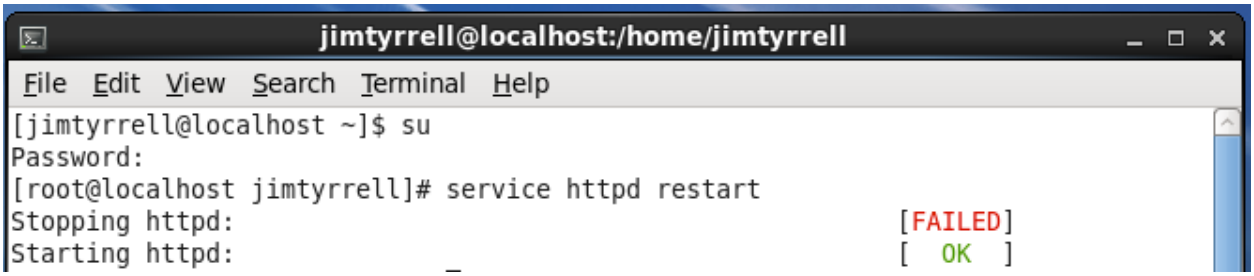
As shown:



Now hit the carriage return so you can see how many lines of output were created when you start httpd:



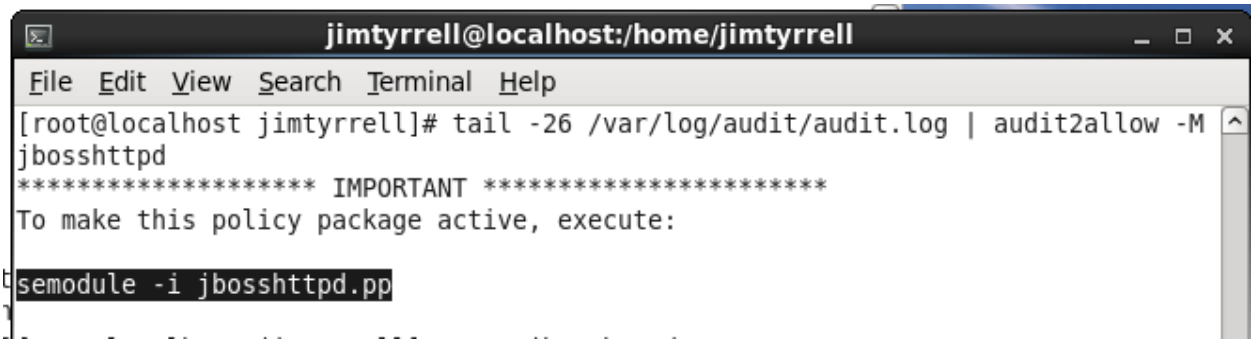
Leave this window running, and open a new window logging in as root, and then running service httpd restart as shown:



You will now see 10 or more lines of output as shown, make sure you clearly count the number of type= lines since you tried to start httpd.

Once you have that number you can run the following command as root:

```
tail -somenumber /var/log/audit/audit.log | audit2allow -M jbosshttpd
```



Now you need to load the module, turn on SE Linux to Enforcing, and then restart httpd and you now have enabled SE Linux with mod_cluster and httpd.