Big Data Cloudera Manager Add New Node to Existing Cluster

Contents

[1 Add New Node to Existing Cluster 2](#_Toc369685940)

[1.1 Scenarios 2](#_Toc369685941)

[1.2 Setup new machines Kerberos / Keytabs 2](#_Toc369685942)

[1.3 Login to Cloudera Manager 2](#_Toc369685943)

[1.4 Request the SSL certificates 7](#_Toc369685944)

[1.5 Create the PKCS12 keystore 7](#_Toc369685945)

[1.6 Convert PKCS12 keystore to JAVA keystore 7](#_Toc369685946)

[1.7 Create the JAVA keystore 8](#_Toc369685947)

[1.8 Repeat for all proxy machines 8](#_Toc369685948)

[1.9 Copy CA Root Certificate to all agent machines. 8](#_Toc369685949)

[1.10 Edit the agent configuration file on all agent machines. 8](#_Toc369685950)

[1.11 Adjust HDFS service roles 9](#_Toc369685951)

[1.12 Adjust Map/Reduce service roles 14](#_Toc369685952)

[1.13 Switch Cloudera Manager to HTTPS/SSL 17](#_Toc369685953)

[1.14 Restart Cloudera Manager server and agents 17](#_Toc369685954)

[2 APPENDIX: keytool / openssl common tasks 18](#_Toc369685955)

[2.1 Java Keytool Commands for Creating and Importing 18](#_Toc369685956)

[2.2 Java Keytool Commands for Checking 18](#_Toc369685957)

[2.3 Other Java Keytool Commands 19](#_Toc369685958)

[2.4 General OpenSSL Commands 19](#_Toc369685959)

[2.5 Checking Using OpenSSL 20](#_Toc369685960)

[2.6 Debugging Using OpenSSL 20](#_Toc369685961)

[2.7 Converting Using OpenSSL 21](#_Toc369685962)

# Add New Node to Existing Cluster

The goal of this document is to provide the process of adding a new machine to an existing cluster. The process involves logging into Cloudera Manager and going through the add host wizard. One challenge is addressing issues with pbrun policy issues. Once you complete this process, a new node will be added to the cluster.

## Scenarios

There are three known scenarios for adding a new host to an existing cluster. Each scenario requires a different approach for handling the software installation and dealing with existing data that might exist on the machine.

**Scenario 1:**  New / clean host: This case involves a host that has never had Cloudera Manager Software installed nor does the host contain any data (/data/\* directories). All that needs to be done is the steps below for adding the host to the existing cluster

**Scenario 2:** Previously used host with existing data and incorrect version of Cloudera Manager Software. The data will need to be deleted via the following commands:

1. “find /data –type d –name dfs –exec rm {} \;”
2. “find /data –type d –name mapred –exec rm {} \;”

Also, the software will need to be removed with the following command:

1. “sudo yum erase cloudera\*”
2. yum clean all
3. yum remove cyrus-sasl-gssapi.x86\_64 fuse.x86\_64 fuse-libs.x86\_64 libxslt.x86\_64
4. rm /etc/yum.repos.d/soe-bigdata.repo

After completing the clean-up, then proceed with the steps below for adding the host to the existing cluster.

## Setup new machines Kerberos / Keytabs

The first task is to request Kerberos principles for the various roles for the new machines. Follow the instructions of sections 1.1.3 and 1.1.4 in the Kerberos document located here: <https://catecollaboration.citigroup.net/domains/deveng/debi/BigData/Cloudera/TechDocs/KERBEROS_for_Cloudera_Manager_and_CDH.pdf>

Make sure to place the new keytabs on the machine running Cloudera Manager Server in the /opt/Cloudera/keytabs directory with the correct ownership and permissions per the Kerberos document.

## Manually Install Cloudera Manager Agent on new host

**NOTE 1: Due to a temporary issue within Cloudera Manager 4.7.2, the Cloudera Manager Agent software must be installed manually.**

**NOTE 2: Please confirm that the latest version of JAVA is installed on the node. If not, please follow the steps in the JAVA upgrade document.** [**https://catecollaboration.citigroup.net/domains/deveng/debi/BigData/Documents/Engineered-Solutions/Cloudera-Hadoop/BigData\_CM\_Add\_New\_Node\_To\_Existing\_Cluster.docx**](https://catecollaboration.citigroup.net/domains/deveng/debi/BigData/Documents/Engineered-Solutions/Cloudera-Hadoop/BigData_CM_Add_New_Node_To_Existing_Cluster.docx)

For each new host, run the installation command as root user:

1. Add the repo files  
   “yum -y install BD\_bigdata\_repo”
2. C4 & C5 All Nodes

pbrun yum -y install fuse redhat-lsb fuse-libs --enablerepo='soe6u4, soe6products, soe-bigdata-c5';

pbrun yum -y install cyrus-sasl-gssapi libxslt --enablerepo='soe6u4, soe6products, soe-bigdata-c5';

1. For C4 All Nodes

pbrun yum install cloudera-manager-daemons cloudera-manager-agent enterprise-debuginfo --enablerepo=soe-bigdata-cm;

1. C4 CM Admin Nodes

pbrun yum install cloudera-manager-server --enablerepo=soe-bigdata-cm;

pbrun yum install BD\_cloudera\_cdh\_parcel --enablerepo=soe-bigdata-cm;

pbrun yum install BD\_cloudera\_cdh\_parcel\* BD\_cloudera\_sentry\_parcel\* BD\_cloudera\_impala\_parcel\* BD\_cloudera\_SOLR\_parcel\*--enablerepo=soe-bigdata-cm

1. C5 All Nodes

pbrun yum install cloudera-manager-daemons cloudera-manager-agent enterprise-debuginfo --enablerepo=soe-bigdata-5;

1. C5 CM Admin Nodes

pbrun yum install cloudera-manager-server --enablerepo=soe-bigdata-5;

pbrun yum install BD\_cloudera\_cdh\_parcel --enablerepo=soe-bigdata-5;

pbrun yum install BD\_cloudera\_cdh\_parcel\* --enablerepo=soe-bigdata-5;

1. Copy the CA root certificate file every machine within the cluster into the directory and file. Also, make the file ownership **root:root** with permission of “**600**” : **/etc/cloudera-scm-agent/ca-root-cert.pem**
2. Edit the following file (**/etc/cloudera-scm-agent/config.ini**) on all machines within the cluster to contain the following two lines. Also, restart the cloudera-scm-agent service after editing the file.

**use\_tls=1**

**verify\_cert\_file=/etc/cloudera-scm-agent/ca-root-cert.pem**

**server\_host=<cm\_server\_host|vip\_dns\_name>**

**NOTE:** The entry: server\_host=<cm\_server\_host> must be changed to use the VIP DNS name if the VIP is used for SAML integration.

**IMPORTANT NOTE:** If the VIP does not forward port 7182, each host running Cloudera Manager Agent can have /etc/hosts and /etc/host.conf updated to make the VIP DNS name resolve to the Cloudera Manager Server host (typically proxy 1).

/etc/hosts:

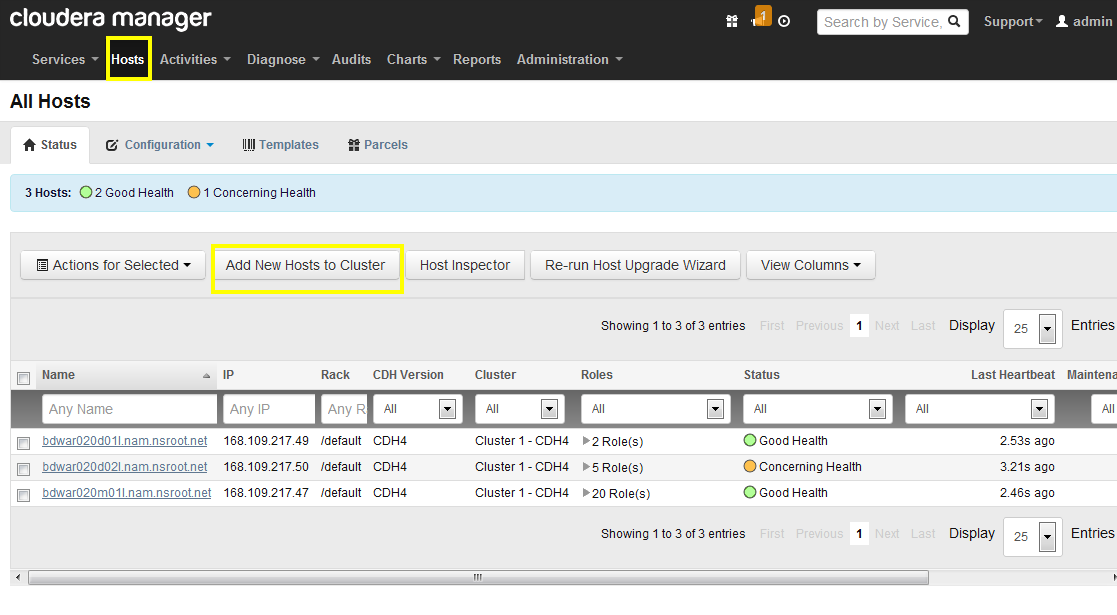
* 1. <ip\_of\_cm\_server> <vip\_dns\_name>
  2. Ex: 10.1.1.1 bigdataplatform.nam.nsroot.net bigdataplatform

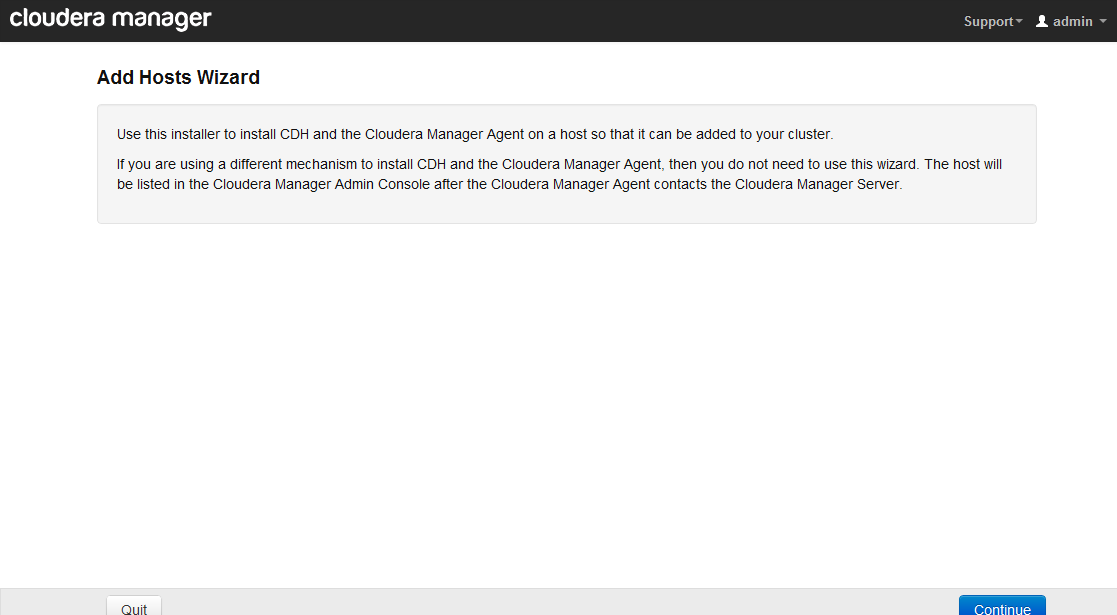
/etc/host.conf:

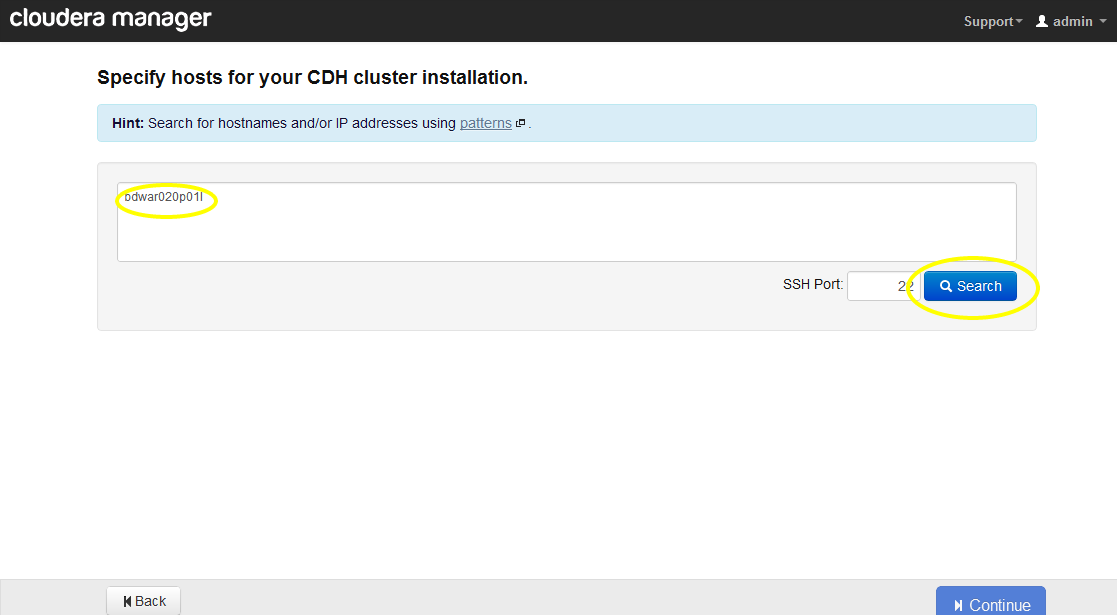
* 1. server=local,bind

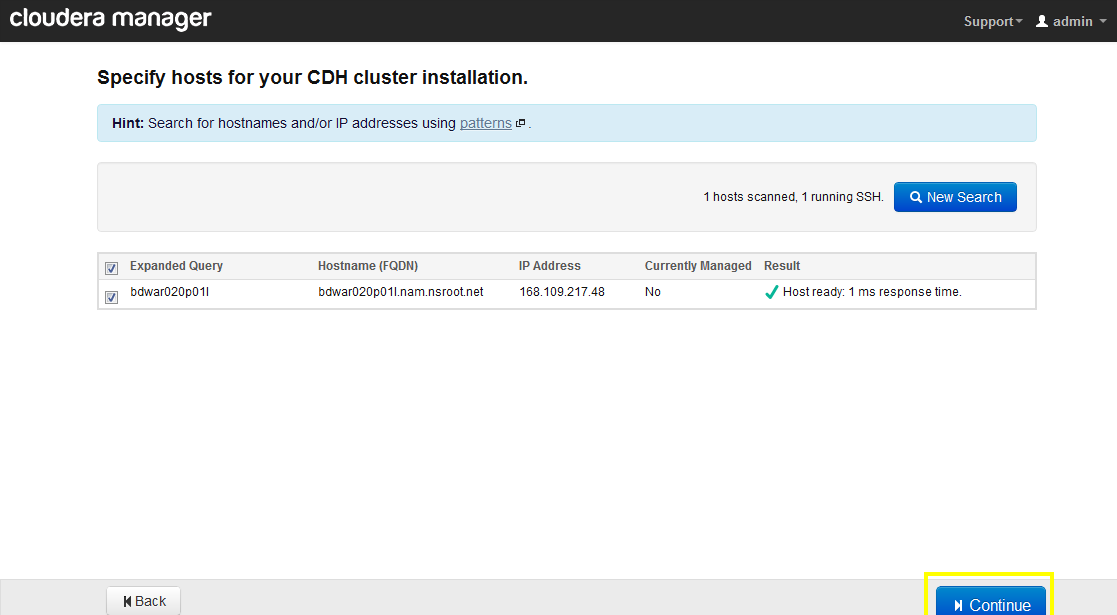
## Login to Cloudera Manager

Login to Cloudera Manager and click on Hosts menu item:

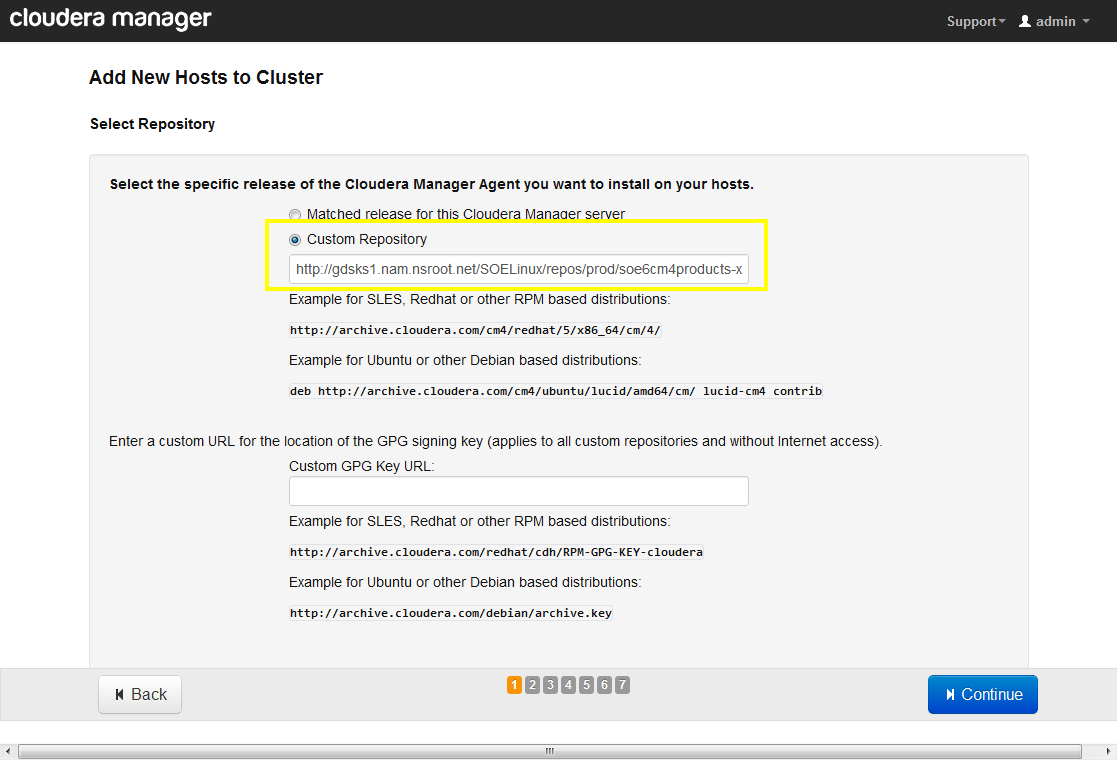




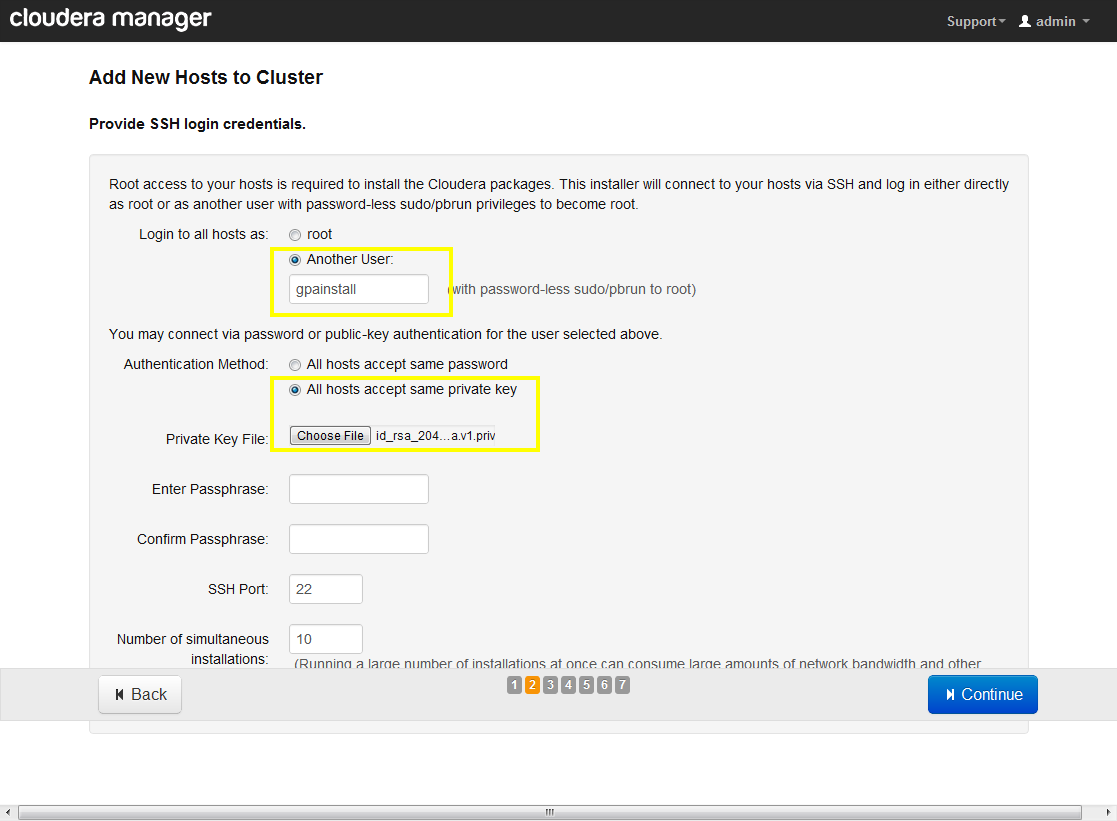


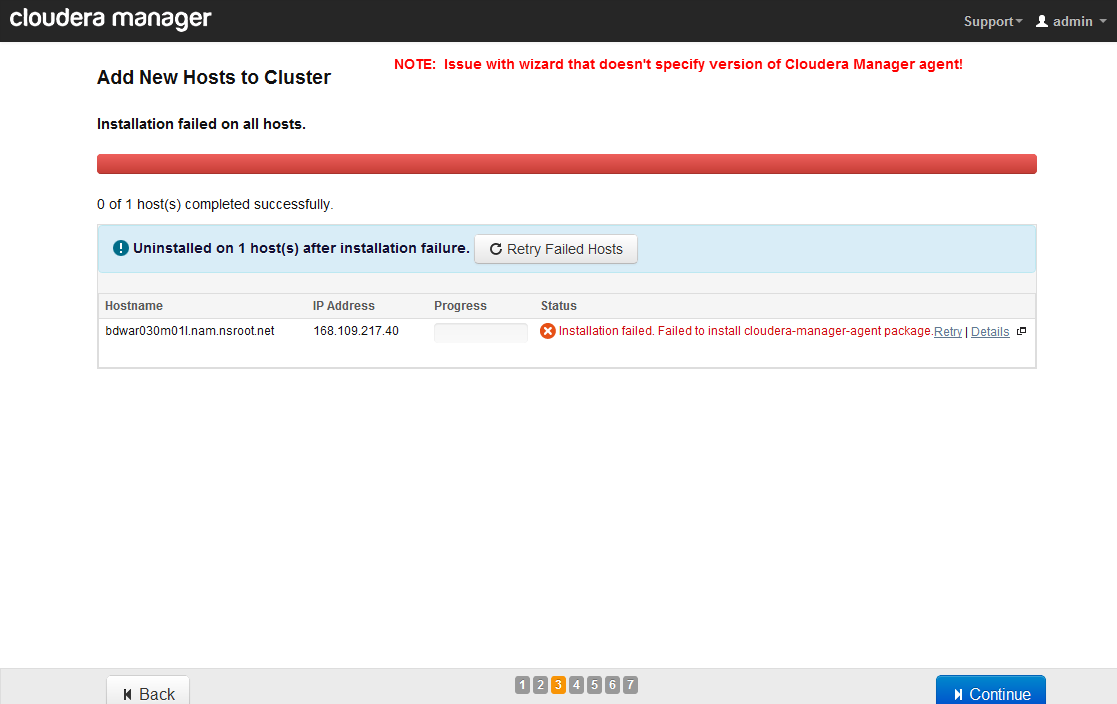


Custom Repository: <http://gdsks1.nam.nsroot.net/SOELinux/repos/prod/soe6cm4products-x86_64/>



Private key file:





## Request the SSL certificates

Obtain the SSL certificates for all of the proxy machines. The ZIP file containing the SSL certificate will also contain the private key and the CA root certificate. Place these items on a unix/linux machine that has openssl and Oracle JAVA (JRE OR JDK).

## Create the PKCS12 keystore

Run the following command:

**openssl pkcs12 -export -inkey <private.key> -in <all-certs.pem> -out file.pkcs12**

**NOTE: <private.key> = the file containing the private key that came with the SSL certificate**

**<all-certs.pem> = the combined CA root certificate and the SSL certificate**

**NOTE: You will be prompted for the private key existing passphrase and a new passphrase for the keystore/pkcs12 file. Please make note of the new passphrase. You will need the passphrase later in this document.**

## Convert PKCS12 keystore to JAVA keystore

Run the following command to produce the keystore:

**/<path\_to\_Oracle\_Java>/jre/bin/keytool -importkeystore -srckeystore file.pkcs12 -srcstoretype PKCS12 -destkeystore keystore**

## Create the JAVA keystore

Copy the keystore to the specific host the SSL certificate is for into the directory with file ownership of **root:root** and permissions **644**:

**/etc/cloudera-scm-server/keystore**

## Repeat for all proxy machines

Repeat step 1.1.3 thru step 1.1.5 for all proxy machines that will run Cloudera Manager

## Copy CA Root Certificate to all agent machines.

Copy the CA root certificate file every machine within the cluster into the directory and file. Also, make the file ownership **root:root** with permission of “**600**” : **/etc/cloudera-scm-agent/ca-root-cert.pem**

## Edit the agent configuration file on all agent machines.

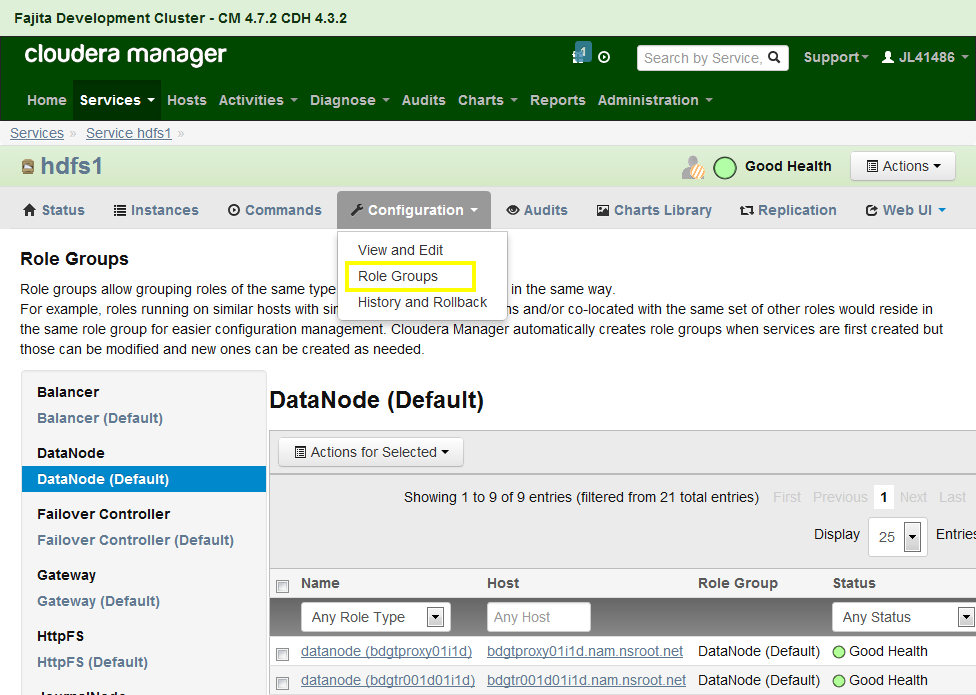
Edit the following file (**/etc/cloudera-scm-agent/config.ini**) on all machines within the cluster to contain the following two lines. Also, restart the cloudera-scm-agent service after editing the file.

**use\_tls=1**

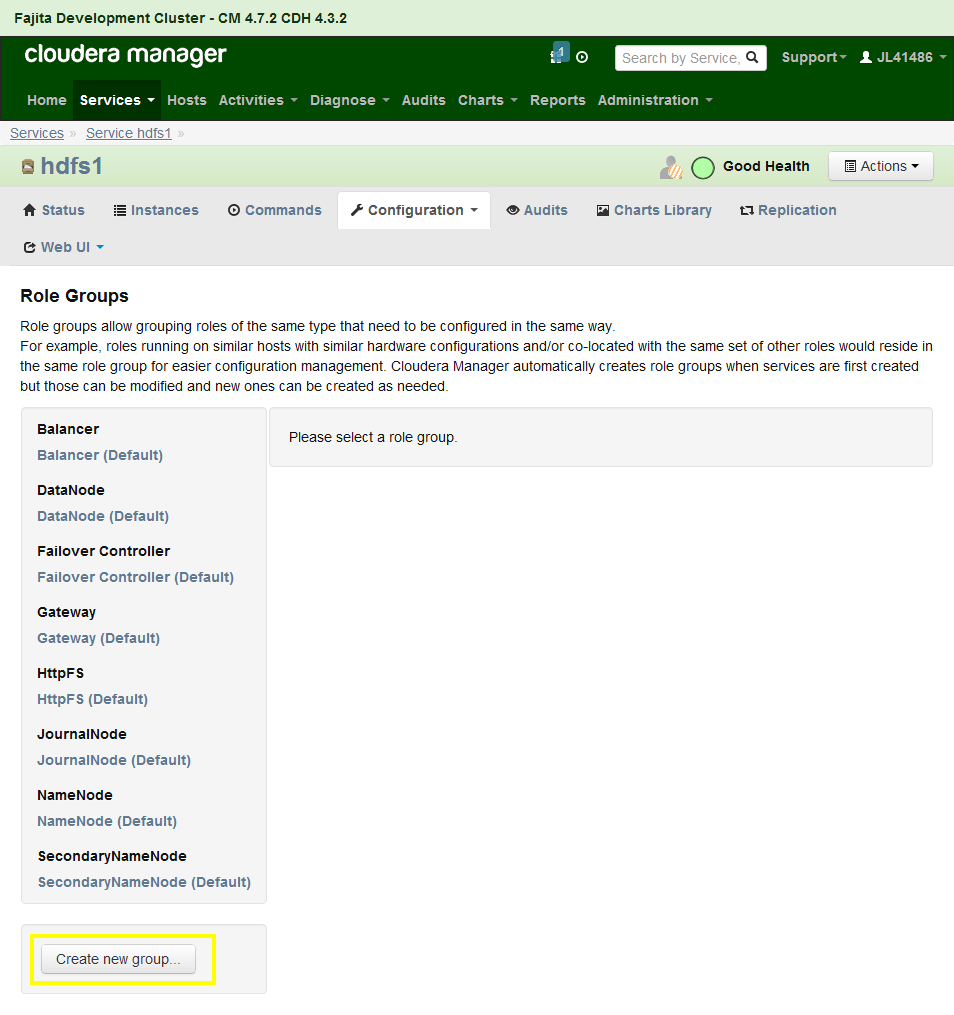
**verify\_cert\_file=/etc/cloudera-scm-agent/ca-root-cert.pem**

## Adjust HDFS service roles

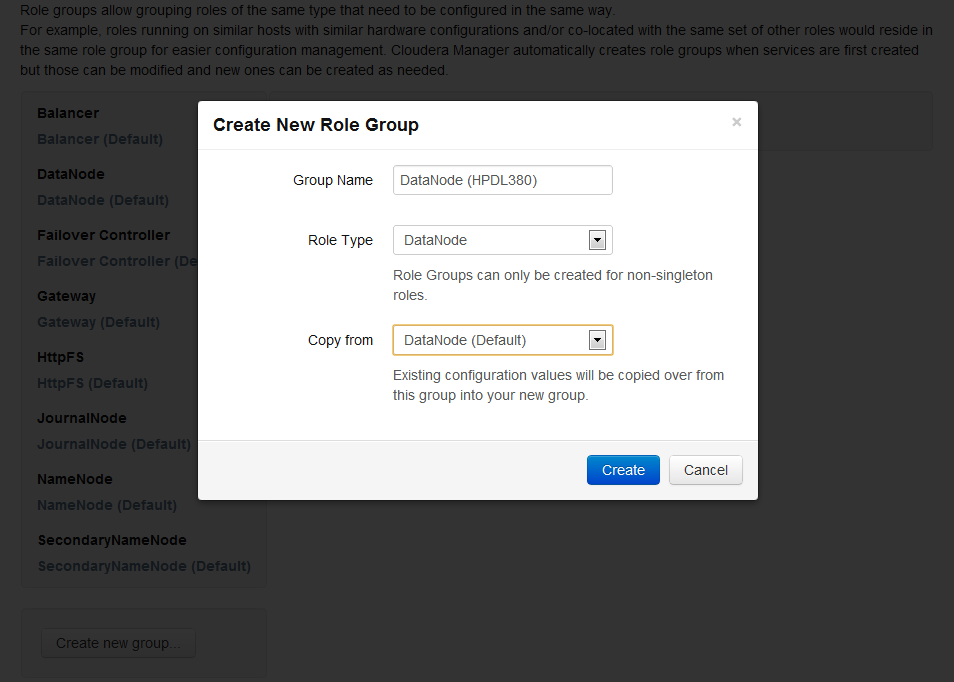
When a cluster is composed of multiple hosts with different hardware models with different CPU core count and hard drive count, then, the hdfs service role groups must be updated. For each unique disk drive count host type, a unique data node role group must be defined. See below for details:



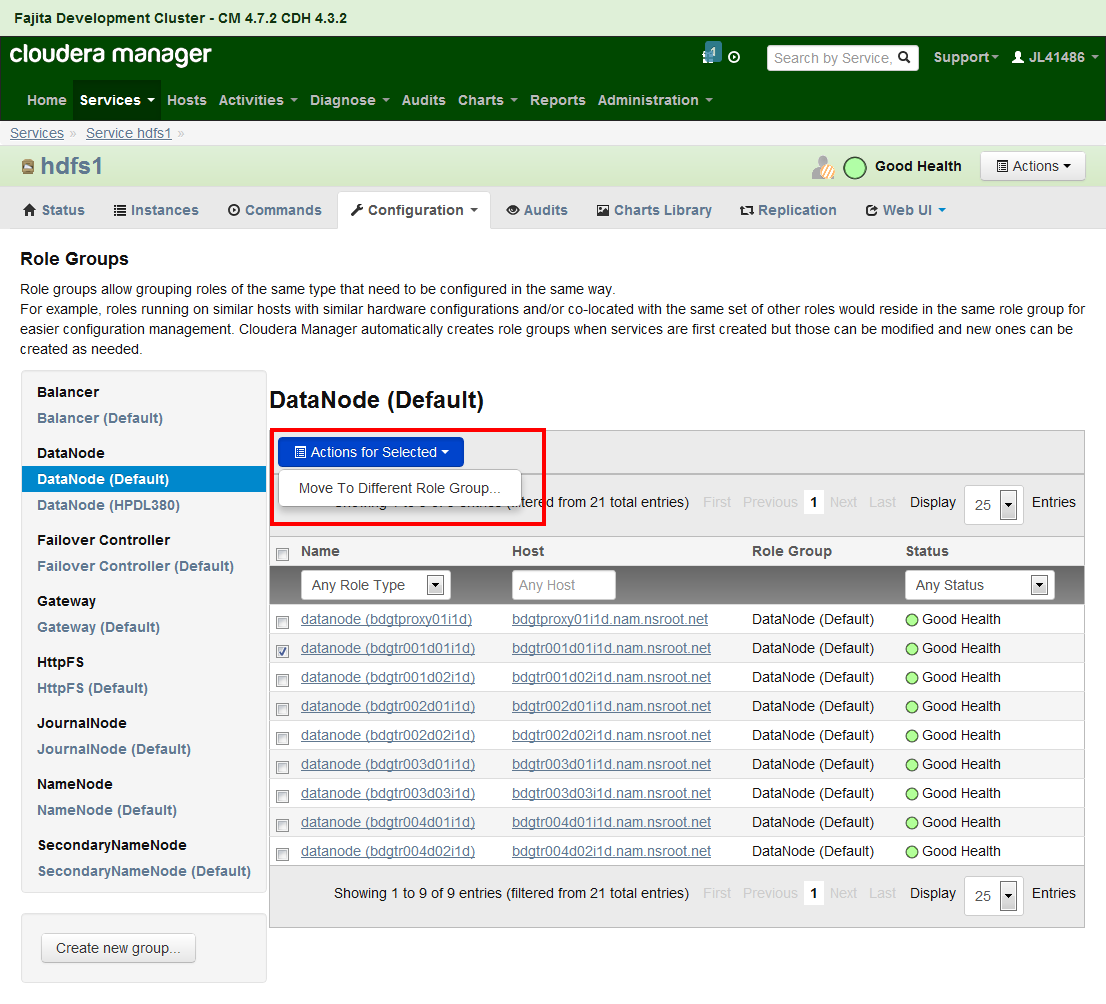
Click on the create new group button:



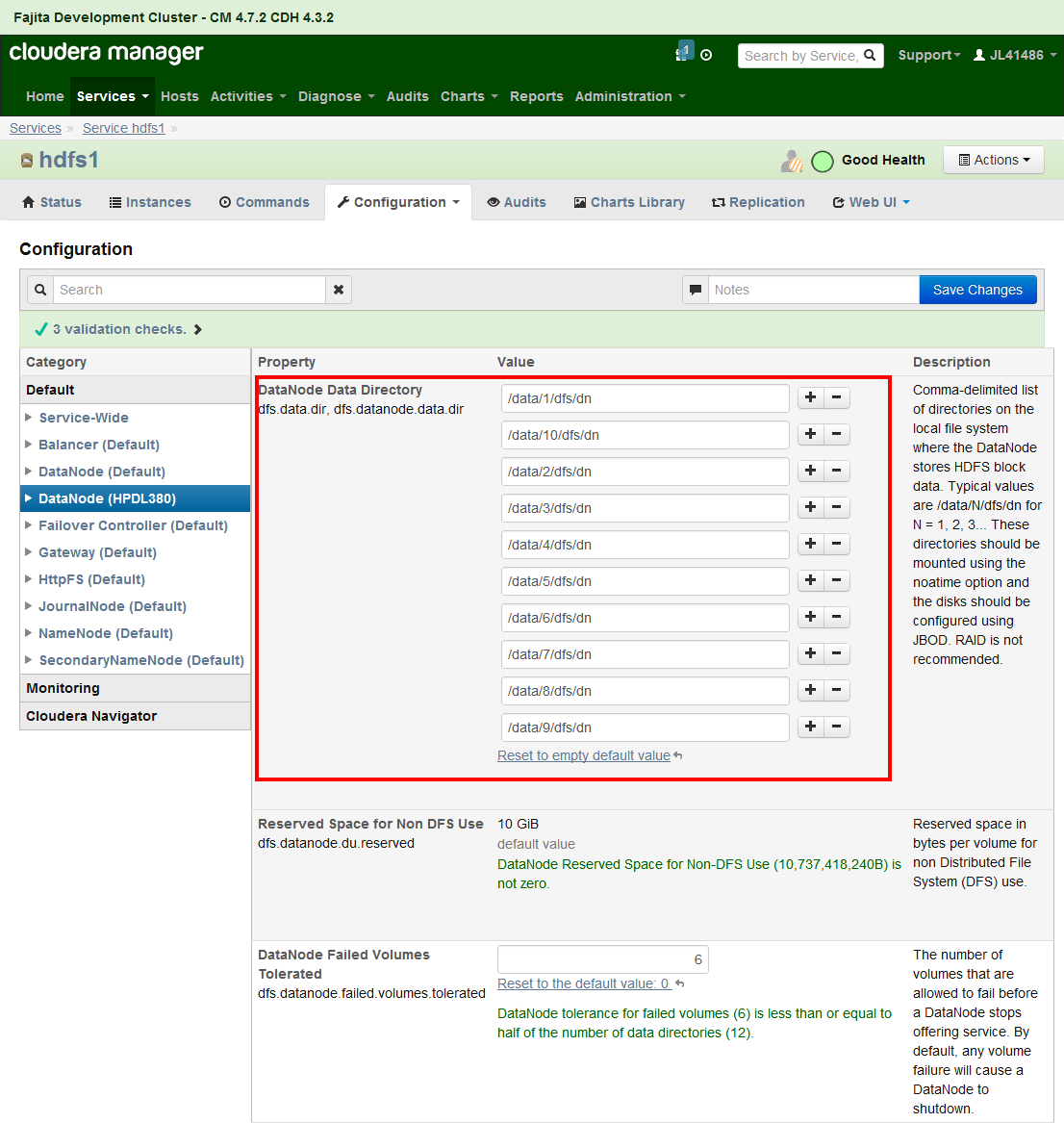
Then, enter a name for the group name which includes some reference to the specific hardware. And copy values from the DataNode (Default) group.



After creating the new role group, the appropriate hosts may need to be moved from the existing role map into the newly created role map. To move hosts, select the existing role map on the left hand side navigation. Then, select all of the hosts that need to be moved and under Actions, select Move item.

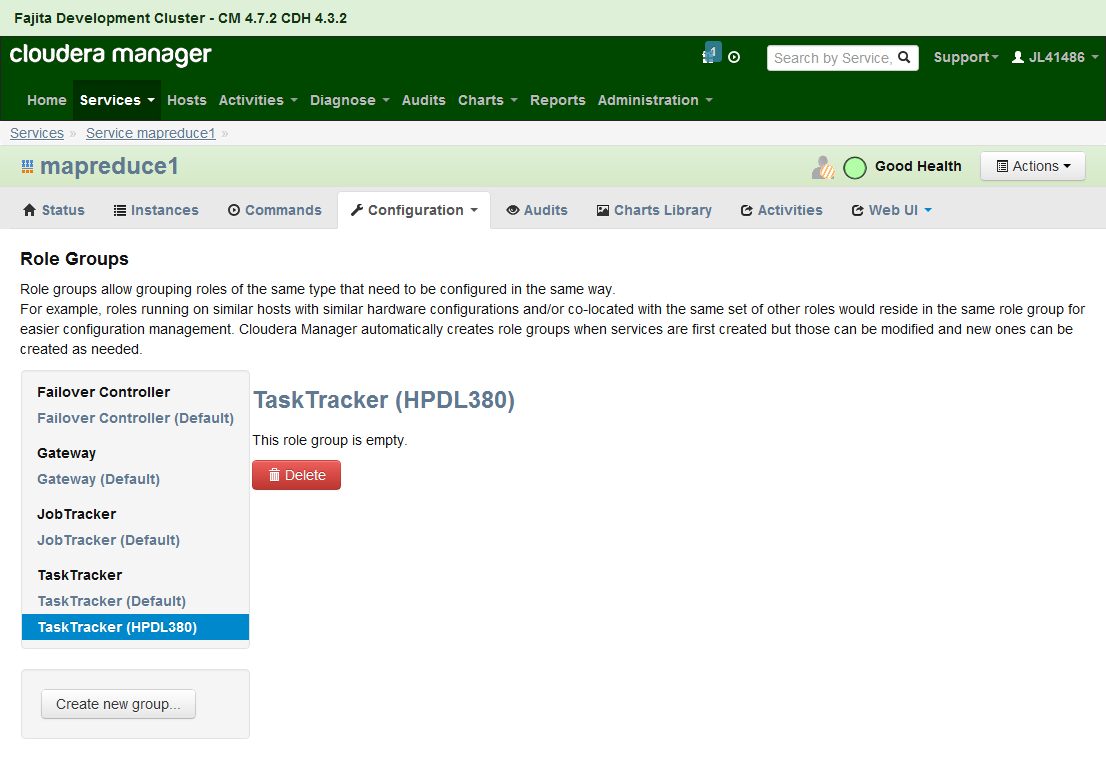


Click on the Configuration button in order to configure the newly created role map. Then, select the newly created role group and edit the data directories.

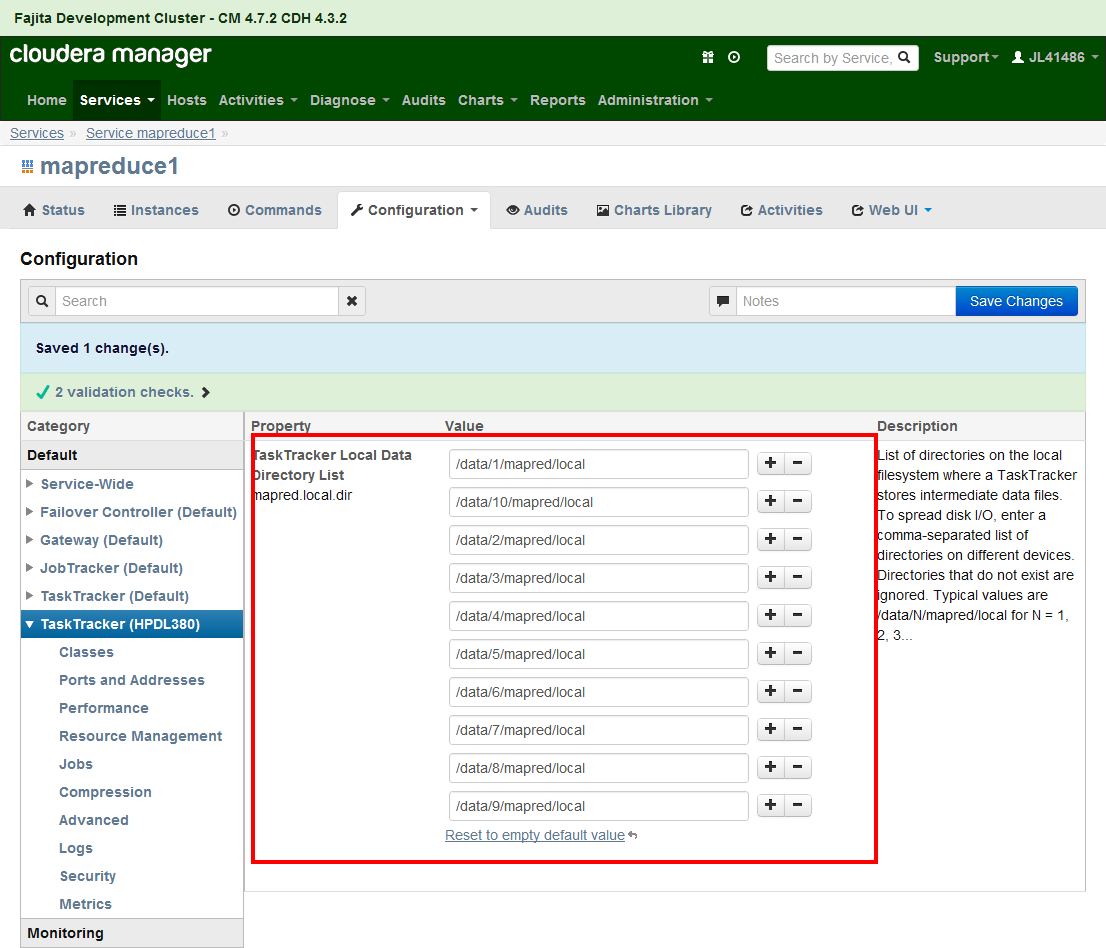


## Adjust Map/Reduce service roles

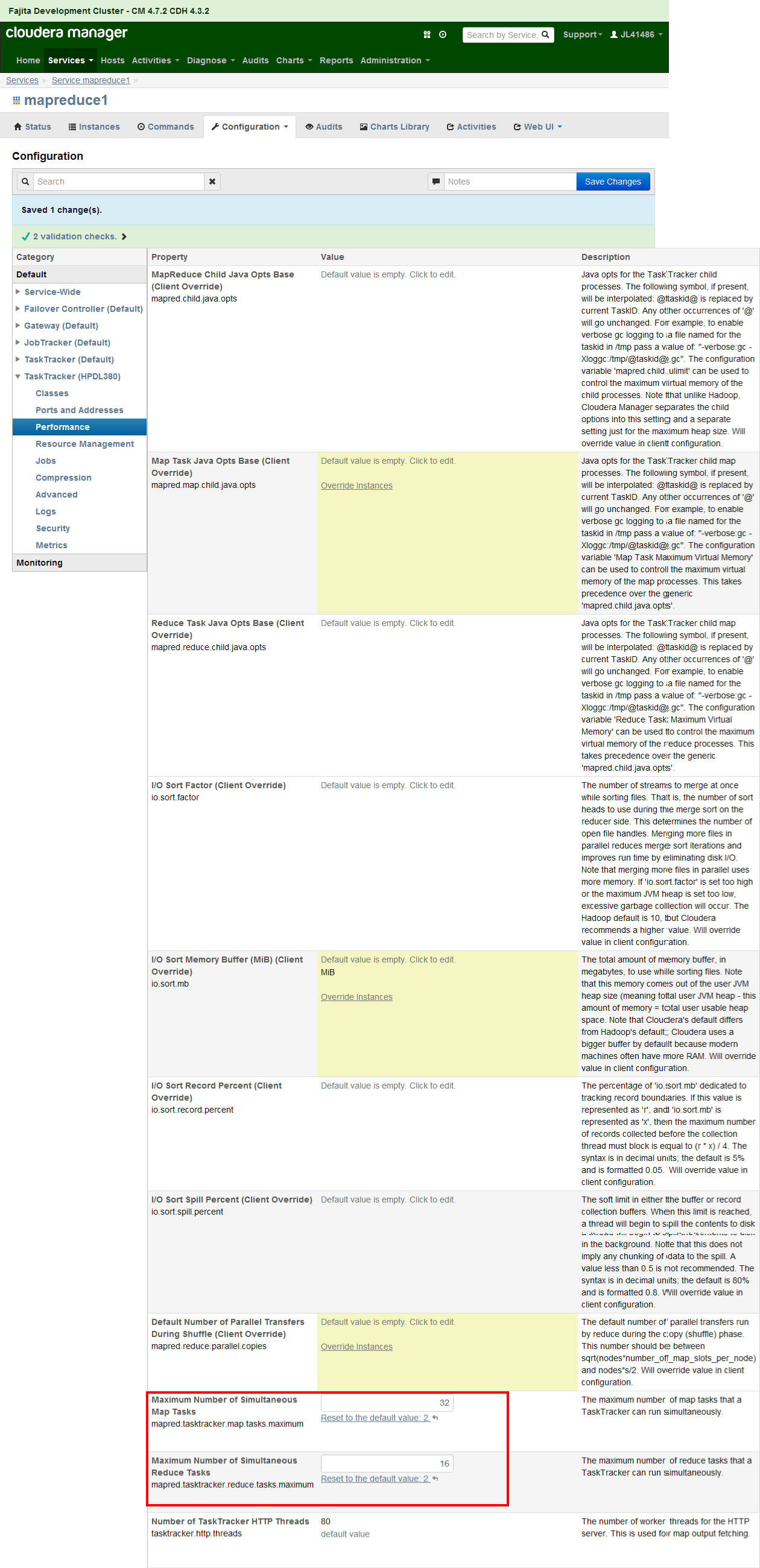
A new role map needs created for the unique hardware under the MapReduce service. The outcome should be something like:



Adjust the new role map’s configuration to contain the appropriate data drives



Also, adjust the max slots for map tasks and reduce tasks

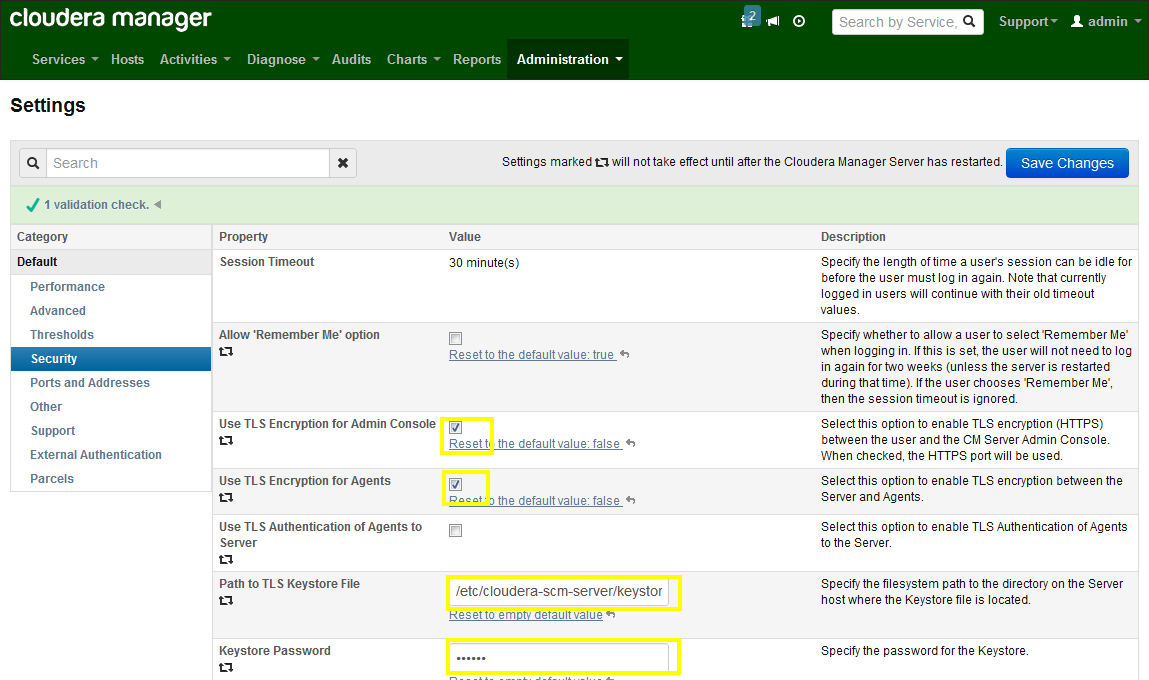


## Switch Cloudera Manager to HTTPS/SSL

Login to Cloudera Manager and under the **Administration** tab select **Settings**. Enter the following as highlighted. Then, click the “Save Changes” button

NOTE: “Path to TLS Keystore file” needs to be: **/etc/cloudera-scm-server/keystore**

NOTE: “Keystore Password” needs to be the keystore passphrase you created in previous step



## Restart Cloudera Manager server and agents

Restart the cloudera-scm-server service. Then, login to Cloudera Manager web application and click on the “Hosts” tab to verify that all agents have reported into and are healthy. Also, the URL of Cloudera Manager should now be HTTPS indicating access via SSL encryption.

You have completed the updating of Cloudera Manager to make all communications encrypted.

# APPENDIX: keytool / openssl common tasks

## Java Keytool Commands for Creating and Importing

These commands allow you to generate a new Java Keytool keystore file, create a CSR, and import certificates. Any root or intermediate certificates will need to be imported before importing the primary certificate for your domain.

* **Generate a Java keystore and key pair**

keytool -genkey -alias mydomain -keyalg RSA -keystore keystore.jks -keysize 2048

* **Generate a certificate signing request (CSR) for an existing Java keystore**

keytool -certreq -alias mydomain -keystore keystore.jks -file mydomain.csr

* **Import a root or intermediate CA certificate to an existing Java keystore**

keytool -import -trustcacerts -alias root -file Thawte.crt -keystore keystore.jks

* **Import a signed primary certificate to an existing Java keystore**

keytool -import -trustcacerts -alias mydomain -file mydomain.crt -keystore keystore.jks

* **Generate a keystore and self-signed certificate** (see [How to Create a Self Signed Certificate using Java Keytool](http://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-using-java-keytool.html) for more info)

keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -validity 360 -keysize 2048

## Java Keytool Commands for Checking

If you need to check the information within a certificate, or Java keystore, use these commands.

* **Check a stand-alone certificate**

keytool -printcert -v -file mydomain.crt

* **Check which certificates are in a Java keystore**

keytool -list -v -keystore keystore.jks

* **Check a particular keystore entry using an alias**

keytool -list -v -keystore keystore.jks -alias mydomain

## Other Java Keytool Commands

* **Delete a certificate from a Java Keytool keystore**

keytool -delete -alias mydomain -keystore keystore.jks

* **Change a Java keystore password**

keytool -storepasswd -new new\_storepass -keystore keystore.jks

* **Export a certificate from a keystore**

keytool -export -alias mydomain -file mydomain.crt -keystore keystore.jks

* **List Trusted CA Certs**

keytool -list -v -keystore $JAVA\_HOME/jre/lib/security/cacerts

* **Import New CA into Trusted Certs**

keytool -import -trustcacerts -file /path/to/ca/ca.pem -alias CA\_ALIAS -keystore $JAVA\_HOME/jre/lib/security/cacerts

If you need to move a certificate from Java Keytool to Apache or another type of system, check out these instructions for [converting a Java Keytool keystore using OpenSSL](http://www.sslshopper.com/article-move-your-java-keytool-ssl-certificate-to-openssl.html). For more information, check out the[Java Keytool documentation](http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html) or check out our [Tomcat SSL Installation Instructions which use Java Keytool](http://www.sslshopper.com/tomcat-ssl-installation-instructions.html).

## General OpenSSL Commands

These commands allow you to generate CSRs, Certificates, Private Keys and do other miscellaneous tasks.

* **Generate a new private key and Certificate Signing Request**

openssl req -out CSR.csr -new -newkey rsa:2048 -nodes -keyout privateKey.key

* **Generate a self-signed certificate (see**[**How to Create and Install an Apache Self Signed Certificate**](http://www.sslshopper.com/article-how-to-create-and-install-an-apache-self-signed-certificate.html)**for more info)**

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout privateKey.key -out certificate.crt

* **Generate a certificate signing request (CSR) for an existing private key**

openssl req -out CSR.csr -key privateKey.key -new

* **Generate a certificate signing request based on an existing certificate**

openssl x509 -x509toreq -in certificate.crt -out CSR.csr -signkey privateKey.key

* **Remove a passphrase from a private key**

openssl rsa -in privateKey.pem -out newPrivateKey.pem

## Checking Using OpenSSL

If you need to check the information within a Certificate, CSR or Private Key, use these commands. You can also [check CSR](http://www.sslshopper.com/csr-decoder.html)s and [check certificates](http://www.sslshopper.com/certificate-decoder.html) using our online tools.

* **Check a Certificate Signing Request (CSR)**

openssl req -text -noout -verify -in CSR.csr

* **Check a private key**

openssl rsa -in privateKey.key -check

* **Check a certificate**

openssl x509 -in certificate.crt -text -noout

* **Check a PKCS#12 file (.pfx or .p12)**

openssl pkcs12 -info -in keyStore.p12

## Debugging Using OpenSSL

If you are receiving an error that the private doesn't match the certificate or that a certificate that you installed to a site is not trusted, try one of these commands. If you are trying to verify that an SSL certificate is installed correctly, be sure to check out the [SSL Checker](http://www.sslshopper.com/ssl-checker.html).

* **Check an MD5 hash of the public key to ensure that it matches with what is in a CSR or private key**

openssl x509 -noout -modulus -in certificate.crt | openssl md5  
openssl rsa -noout -modulus -in privateKey.key | openssl md5  
openssl req -noout -modulus -in CSR.csr | openssl md5

* **Check an SSL connection. All the certificates (including Intermediates) should be displayed**

openssl s\_client -connect www.paypal.com:443

## Converting Using OpenSSL

These commands allow you to convert certificates and keys to different formats to make them compatible with specific types of servers or software. For example, you can convert a normal PEM file that would work with Apache to a PFX (PKCS#12) file and use it with Tomcat or IIS. Use our [SSL Converter to convert certificates](https://www.sslshopper.com/ssl-converter.html) without messing with OpenSSL.

* **Convert a DER file (.crt .cer .der) to PEM**

openssl x509 -inform der -in certificate.cer -out certificate.pem

* **Convert a PEM file to DER**

openssl x509 -outform der -in certificate.pem -out certificate.der

* **Convert a PKCS#12 file (.pfx .p12) containing a private key and certificates to PEM**

openssl pkcs12 -in keyStore.pfx -out keyStore.pem -nodes

You can add -nocerts to only output the private key or add -nokeys to only output the certificates.

* **Convert a PEM certificate file and a private key to PKCS#12 (.pfx .p12)**

openssl pkcs12 -export -out certificate.pfx -inkey privateKey.key -in certificate.crt -certfile CACert.crt