Hortonworks Data Platform

System Administration Guides

(Apr 2, 2013)

docs.hortonworks.com

Hortonworks Data Platform : System Administration Guides

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

List of Figures

List of Tables

1. High Availability for Hadoop

This section provides information on the various components of the Apache Hadoop ecosystem and setting them up for high availability.

HDPs Full-Stack HA Architecture

Hortonworks Data Platform (HDP) is an open source distribution powered by Apache Hadoop. HDP provides you with the actual Apache-released versions of the stack with all the necessary bug fixes to make all the components in the stack interoperable in your production environments. This stack uses multiple 'master' services whose failure would cause functionality outage in your Hadoop cluster. Hortonworks' Full-Stack High Availability architecture provides a common framework to make all the master services resilient to failures.

HDP uses industry proven HA technologies in order to provide a reliable HA solution.

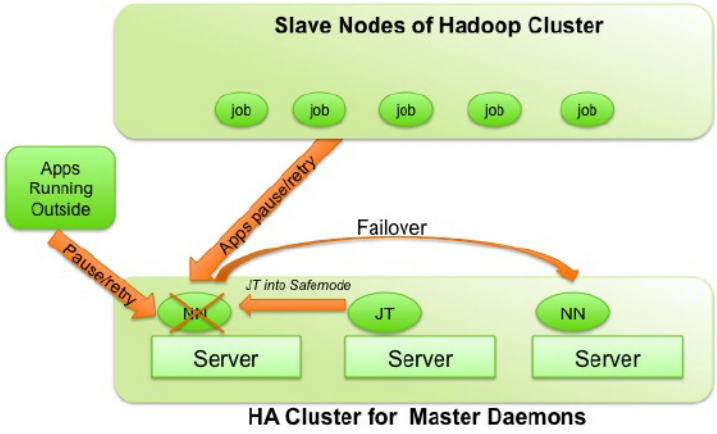
The Hadoop stack contains multiple services (HDFS, MapReduce, HBase, etc.) and each of these services have their own co-dependencies. A client application, that interacts with Hadoop, can depend on one or more of these services. A highly available Hadoop platform must ensure that the NameNode master service as well as client applications are resilient to critical failure services. Hortonworks' Full-Stack HA architecture considers this global view.

Also see. Hortonworks blog on NameNode HA with Hadoop 1.0 The HDP HA architecture has the following key properties:

- It provides high availability for the NameNode master daemon service.
- When the NameNode master daemon fails over, the HA solution initiates the following actions:
  - Dependent services (like JobTracker) automatically detect the failure or fail over of the co-dependent component (NameNode) and these dependent services pause, retry, and recover the failed service. (For example, the JobTracker does not launch new jobs or kill jobs that have been waiting for the NameNode.)
  - Applications running inside and outside the Hadoop cluster also automatically pause and retry their connection to the failed service.

The above actions are highlighted in the following illustration. This illustration shows how HDFS clients and MapReduce services (Jobtracker daemon) handle the NameNode fail over.

Figure 1.1. FullStackHA.jpg



To configure High Availability for your Hadoop cluster see:

- High Availability using Red Hat
- High Availability using VMWare

2. High Availability for HDFS NameNode Using VMware

This section describes using VMWare to support high availability.

2.1. High Availability for Hadoop Using VMware

This document is intended for system administrators for configuring the Hadoop NameNode (NN) service for High Availability and also for configuring clients and other Hadoop services to be resilient while the NameNode fails over.

The Hortonworks Data Platform (HDP) High Availability (HA) kit for VMware provides enterprise grade high availability of the NameNode Service as part of the HDP cluster.

To learn more, see HDP's Full-Stack HA Architecture.

In this document:

- Use Cases and Fail Over Scenarios
- Supported Operating Systems
- Configuration For Physical Servers
- Software Configuration
- Enable vSphere for HA
- Validate NameNode High Availability
- Administration Guide for Highly Available NameNode
- References

2.2. Use Cases and Fail Over Scenarios

This section provides information on the use cases and fail over scenarios for high availability in Hadoop.

Use Cases

This solution enables the following Hadoop system administration use cases:

- Planned failure of the NameNode (for maintenance tasks like software or hardware upgrade)
- Unplanned failure of the NameNode

Fail over scenarios

The solution deals with the following faults:

- NameNode service failure
- NameNode JVM failure
- Hung NameNode daemon or hung operating system
- NameNode operating system failure
- Virtual machine failure
- ESXi host failure
- Failure of the NIC cards on ESXi hosts.
- Network failure between ESXi hosts.

Note

Some double faults are not handled (such as failure of multiple ESXi hosts).

2.3. Supported Operating Systems

The following operating systems are supported:

- Red Hat Enterprise Linux (RHEL) v5.7, v5.8, v6.2, v6.3 and later
- CentOS v5.7, v5.8, v6.2, v6.3 and later

2.4. Configuration For Physical Servers

Ensure that your setup meets the following requirements:

- The NameNode must run inside a virtual machine which is hosted on the vSphere HA cluster.
- The vSphere HA cluster must include a minimum of two ESXi server machines. See the NameNode hardware recommendations available here.
- The vSphere HA cluster must use shared storage.

Note

Shared storage stores the state of the NameNode edit journal and checkpointed image. It provides ability to start the NameNode on another physical machine in the vSphere HA Cluster. Additionally, vSphere uses the shared state for the NameNode's root VM disk.

2.5. Software Configuration

Complete the following tasks to configure NameNode HA solution:

- Configure a vSphere HA cluster
- Install HDP
- Configure NameNode for automatic fail over
- Validate autostart configuration

2.5.1. Configure a vSphere HA cluster

Complete the following configurations to configure your vSphere HA cluster:

- Configure vSphere cluster [5]
- Configure shared storage [5]

Configure vSphere cluster

To configure vSphere cluster, use the vSphere Management Console and complete the following instructions. (Also see the VMware vSphere documentation.):

1. Create a vSphere HA cluster.
   - In the vCenter Client, select Inventory -> Hosts and Clusters.
   - In the left column, right-click the Datacenter and select New Cluster.

   This step will create a vSphere HA cluster. (The vSphere HA cluster hosts the NameNode's VM.)
2. Configure VM to host the NameNode
   - Within the vSphere HA cluster, create the virtual machine. (This VM will host the NameNode process.)
   - Store the virtual machine disk (VMDK) on the data store located on the shared storage device.
   - Specify the exact machine hardware configurations as required for configuring a physical server for the NameNode process.

     Use the NameNode hardware recommendations available here.
   - Start this virtual machine and configure the network settings. You should now have the IP address and FQDN for the virtual machine.
   - The virtual machine should be present on the same network where the rest of the nodes of your Hadoop cluster are located.

Configure shared storage

- Shared storage is required to host both the NameNode data directory and the VM storage contents.

- Use standard enterprise practices (such as configuring the shared storage to be RAIDed) to ensure that your shared storage is reliable.

2.5.2. Install HDP

Use the following instructions to install HDP on your cluster hardware. Ensure that you specify the virtual machine (configured in the previous section) as your NameNode.

1. Download Hortonworks Data Platform (HDP) using the instructions provided here.

Note

Do not start the server until you have configured the relevant templates as outlined in the following steps.

2. Add the configuration parameters for Full-Stack HA.

   a. Edit the`<master-install-machine-for-HDP>/etc/puppet/master/modules/hdp-hadoop/templates/hdfs-site.xml.erb` file to add the following properties:

   - Enable the HDFS client retry policy.

```
<property>
 <name>dfs.client.retry.policy.enabled</name>
 <value>true</value>
 <description> Enables HDFS client retry in case of NameNode failure.</
description>
</property>
```

   - Configure protection for NameNode edit log.

```
<property>
 <name>dfs.namenode.edits.toleration.length</name>
 <value>8192</value>
 <description> Prevents corruption of NameNode edit log.</description>
</property>
```

   - Configure safe mode extension time.

```
<property>
 <name>dfs.safemode.extension</name>
 <value>10</value>
 <description> The default value (30 seconds) is applicable for
 very large clusters. For small to large clusters (upto 200 nodes),
 recommended value is 10 seconds.</description>
</property>
```

   - Ensure that the allocated DFS blocks persist across multiple fail overs.

```
<property>
 <name>dfs.persist.blocks</name>
 <value>true</value>
 <description> Ensure that the allocated DFS blocks persist across
 multiple fail overs.</description>
</property>
```

- Configure delay for first block report.

```
<property>
 <name>dfs.blockreport.initialDelay</name>
 <value>10</value>
 <description> Delay (in seconds) for first block report. </
description>
</property>
```

b. Edit the`<master-install-machine-for-HDP>/etc/puppet/master/modules/hdp-hadoop/templates/mapred-site.xml.erb` file to add the following properties:

- Enable the JobTracker's safe mode functionality.

```
<property>
 <name>mapreduce.jt.hdfs.monitor.enable</name>
 <value>true</value>
 <description> Enable the JobTracker to go into safe mode when the
 NameNode is unresponsive.</description>
</property>
```

- Enable retry for JobTracker clients (when the JobTracker is in safe mode).

```
<property>
 <name>mapreduce.jobclient.retry.policy.enabled</name>
 <value>true</value>
 <description> Enable the MapReduce job client to retry job submission
 when the JobTracker is in safe
    mode.</description>
</property>
```

- Enable recovery of JobTracker's queue after it is restarted.

```
<property>
 <name>mapred.jobtracker.restart.recover</name>
 <value>true</value>
 <description>  Enable the JobTracker to recover its queue after
 JobTracker is restarted.</description>
</property>
```

c. Edit the`<master-install-machine-for-HDP>/etc/puppet/master/modules/hdp-hadoop/templates/core-site.xml.erb` file to add the following properties:

- Configure checkpoint interval so that the checkpoint is performed on an hourly basis.

```
<property>
 <name>fs.checkpoint.period</name>
 <value>3600</value>
 <description> The number of seconds between two periodic checkpoints.
</description>
</property>
```

3. Complete HDP installation.

- Continue the installation process using the instructions provided here.

- Ensure that you also follow the instructions listed below:
  - Use the fully qualified domain name (FQDN) of the virtual machine for configuring the host names (see: Installing, Configuring, and Deploying the Cluster - Assign Masters.).

    Note

    HDP might not identify the NameNode VM automatically and it is therefore important to note down FQDN (IP address and DNS name) of the NameNode VM.

  - Specify shared storage for the NameNode's directories (see: HDP (Ambari) - Customize Services).
  - Do not use the NameNode VM for running any other master daemon.
- Complete the HDP installation. Ensure that the installation was successful.

  Note

  To modify the parameters in the template files after you have installed HDP, ensure that you follow the instructions listed below:

  - Change the template files as required.
  - Stop and start the respective service through the Ambari console GUI.

    For example, stop and restart the MapReduce service if the `mapreduce-site.xml.erb` file is modified. Stop and restart HDFS service if either the `core-site.xml.erb` or the `hdfs-site.xml.erb` file is modified.

2.5.3. Configure NameNode for automatic fail over

Complete the following instructions to configure automatic fail over:

- Install and configure vSphere Monitoring Agent [8]
- Configure the NameNode to start automatically on OS boot [9]

Install and configure vSphere Monitoring Agent

The NameNode's monitoring agent monitors the NameNode daemon and notifies the vSphere if the NameNode daemon fails or becomes unstable. As soon as the vSphere receives the notification, it triggers HA solution to restart NameNode VM on either the same or a different ESXi host.

Follow the steps listed below to install and configure the monitoring agent:

1. On the NameNode VM, download the HDP HA kit for VMware.
   - Download the Hortonworks Data Platform High Availability Kit for VMware:

- For RHEL/CentOS 5.x:

```
wget http://public-repo-1.hortonworks.com/HDP/vmw_artifacts/1.2.1/
centos5/hdp-ha-vmw-1.1.0.23.tar.gz
```

```
wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/1.
2.1/hamonitor_vsphere/hmonitor-vsphere-namenode-daemon-1.1.0.23-1.el5.
x86_64.rpm
```

- For RHEL/CentOS 6.x:

```
wget http://public-repo-1.hortonworks.com/HDP/vmw_artifacts/1.2.1/
centos6/hdp-ha-vmw-1.1.0.23.tar.gz
```

```
wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/1.
2.1/hamonitor_vsphere/hmonitor-vsphere-namenode-daemon-1.1.0.23-1.el6.
x86_64.rpm
```

- Install the NameNode monitoring functionality:

```
yum install hmonitor
```

```
rpm -ivh hmonitor-vsphere-monitoring-1.1.0.2*
```

```
yum install hmonitor-vsphere-namenode-daemon
```

2. Configure the monitoring agent to point to the NameNode configurations.

   - Edit the `/usr/lib/hadoop/monitor/vm-namenode.xml` file and provide the FQDN for your NameNode VM here:

```
<property>
 <name>service.monitor.portprobe.host</name>
 <value>$NameNode_FQDN</value>
 <description> Port to probe via a telnet operation.</description>
</property>
```

   - Provide the HDFS port number.

```
<property>
 <name>service.monitor.portprobe.port</name>
 <value>$HDFS_filesystem_port</value>
 <description> Port to probe via a telnet operation.</description>
</property>
```

   - Provide the FQDN and port information for the HDFS filesystem.

```
<property>
 <name>service.monitor.webprobe.url</name>
 <value>$http://NameNode_FQDN:NameNode_web_ui_port/</value>
 <description> URL to get.</description>
</property>
```

Configure the NameNode to start automatically on OS boot

The monitoring agent requires that the NameNode process should start automatically once the virtual machine is bootstrapped.

To configure this, use the following command:

```
yum install hadoop-namenode
```

2.5.4. Validate autostart configuration

Use the following instructions to validate that both the NameNode service and the monitoring agent automatically get started on the reboot of the NameNode machine:

1. Reboot the NameNode machine.
2. Verify that the NameNode service is up and running.
   - On the NameNode machine, execute the following command:

     ```
     service hadoop-namenode status
     ```

   - If the service is up and running, you should see a message similar to the one shown in the following example:

     ```
     namenode (pid 4651) is running...
     ```

3. Verify that the NameNode monitoring agent is up and running.
   - On the NameNode machine, execute the following command:

     ```
     service hmonitor-namenode-monitor status
     ```

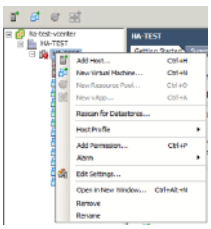
   - If the service is up and running, you should see a message similar to the one shown in the following example:

     ```
     java (pid 4870) is running...
     ```

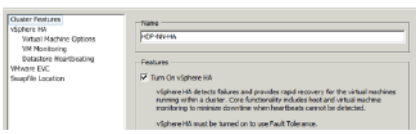
2.5.5. Enable vSphere for HA

Use the following instructions to modify the settings for the vSphere HA cluster created in the previous section:

1. Right click on the cluster in vCenter and select 'Edit Settings'.

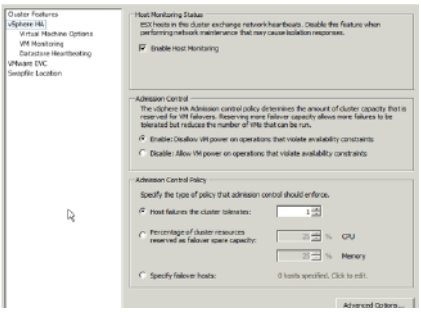


2. In the "Cluster Features" panel, enable the "Turn on vSphere HA" option.

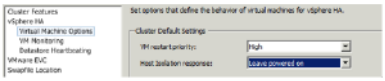


3. In the left navigation menu, select the "vSphere HA" option and configure the following parameters.

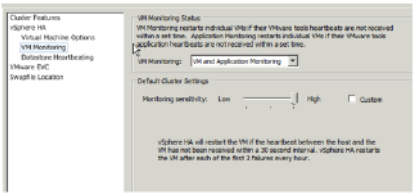
- Enable the “Enable Host Monitoring” option.
- In the same panel, enable Admission Control.
- Under “Admission Control Policy” section, select “Host failures the cluster tolerates” and set the number of failures as 1.



4. In the left navigation menu, select the “Virtual Machine Options” and configure the following parameters.
   - Set the “VM restart priority” to ‘High’.
   - Set the “Host Isolation response” to ‘Leave powered on’.



5. In the left navigation menu, select the “VM Monitoring” option and configure the following parameters:
   - Set the “VM Monitoring” to ‘VM and Application Monitoring’.
   - Set the “Monitoring sensitivity” slider to ‘High’.



2.5.6. Validate NameNode High Availability

It is critical to verify that the NameNode HA infrastructure is functional.

To commission a NameNode HA cluster, we recommend performing explicit tests of vSphere's ability to detect a failed NameNode and to react to that failure. These tests should be performed in addition to any other commissioning tests.

To aid in this commissioning process, the HA monitor RPM includes the Hortonworks Application Monitor (HAM) application. HAM can be used to monitor and restart a

remote NameNode. To validate NameNode High Availability, ensure that you follow the instructions as outlined below:

- Install and configure HAM
- Invoke HAM application
- Validate the failover behavior

2.5.6.1. Install and configure HAM

Typically, HAM can be run on the NameNode. However, we recommend that you also install HAM on other nodes in the cluster because the NameNode will be restarted as part of the validation tests. Use the following command to install HAM:

- For RHEL v5.x:

```
rpm -ivh hmonitor-vsphere-namenode-daemon-1.1.0.23.el5.x86_64.rpm
```

- For RHEL v6.x:

```
rpm -ivh hmonitor-vsphere-namenode-daemon-1.1.0.23.el6.x86_64.rpm
```

2.5.6.2. Invoke HAM application

Use the following command to invoke the HAM application:

```
/usr/lib/hadoop/monitor/ham.sh -conf hadoop-site.xml
```

This displays a GUI that lists the current values and state of the NameNode and Job Tracker.

You can use this GUI to trigger blocking and non-blocking HDFS operations.

Blocking operations will block until the NameNode has failed over. The blocking operations use the `dfs.retry` properties. Non-blocking operations will fail if the NameNode is down, but these operations start succeeding once the failover NameNode is live.

2.5.6.3. Validate the fail over behavior

Use the following tests to verify fail over behavior. (These tests can also be used to verify that the availability monitoring can be suspended for administrative tasks.)

- Verify that NameNode failure triggers the fail over [12]
- Verify that a hung NameNode triggers the fail over [13]
- Verify that ESXi server failure triggers the fail over [14]
- Verify that no fail over is triggered on planned shutdown of the monitor service [14]
- Verify that the monitor provides a bootstrap period before reporting that the NameNode is not live [15]
- Verify that no fail over is triggered when the NameNode enters the safe mode [15]

Verify that NameNode failure triggers the fail over

1. Start the NameNode VM and run the HAM application configured to work with this NameNode.
2. In HAM, start blocking LS operations.
3. SSH to the NameNode VM and terminate the NameNode process.

   ```
   service hadoop-namenode stop
   ```

   Alternatively, identify the NameNode process (**jps -v**) and issue **kill -9** command.
4. Ensure that you see the following expected results:
   - In HAM, the NameNode status area (at the top of the application) should display offline status for NameNode. The main area should also stop displaying any new text (this indicates that the file system operations are blocked).
   - In the vSphere Management UI, the vSphere should terminate the NameNode VM within 60-90 seconds and must start a new instance.
   - Once the NameNode service restarts, its status must be displayed in both the vSphere UI and in the status indicator of HAM.
   - The blocking operations started in HAM must now continue. The fail over should not affect the client except for the pause during fail over.
   - SSH to the NameNode VM again and verify that the host name, IP address, and SSH host key have not changed.

Verify that a hung NameNode triggers the fail over

This test verifies that the VM does not fail immediately after the NameNode process is hung. The monitor considers this time period as a Garbage Collection-related pause. The monitor provides a (configurable) period of grace time before it terminates the hung NameNode process.

1. Start the NameNode VM and run the HAM application configured to work with this NameNode.
2. In HAM, start non-blocking operations.
3. SSH to the NameNode VM and identify the NameNode process.

   ```
   jps -v| grep namenode
   ```
4. Suspend the NameNode process.

   ```
   kill -19 namenode-process-id-here
   ```
5. Ensure that you see the following expected results:
   - In HAM, the NameNode status area must indicate hung status. The non-blocking operations, that are initiated, will now appear to be blocked (the hung NameNode prevents these operations from completing).
   - In the vSphere Management UI, the vSphere should terminate the NameNode VM and start a new instance within a delay of approximately 2-3 minutes.

- In HAM, the NameNode status area should indicate offline status. The non-blocking operations should now report failure.
- Once the NameNode service restarts, its status must be displayed in both the vSphere UI and in the status indicator of HAM.
- The operations started in HAM will now start succeeding.

This test can be repeated when HAM performs blocking operations.

In this case, the active filesystem operation (the operation when the NameNode was suspended) will fail when the NameNode is restarted and reported as such. This failure happens because the open socket connection breaks and these connections are not preserved during a fail over.

Verify that ESXi server failure triggers the fail over

This test verifies that the HA solution detects the failures of the physical hardware and also trigger fail overs.

1. Start the NameNode VM in an ESXi server that is not running any other VMs.
2. Run the HAM application configured to work against this NameNode.
3. In the HAM, start blocking LS operations.
4. Initiate a power down of the ESXi server.
5. Ensure that you see the following expected results:
   - The main area should stop displaying new text - this indicates that the file system operations are blocked.
   - In the vSphere management UI, once the loss of the ESXi server is detected, the NameNode VM is re-instantiated on one of the remaining ESXi servers.
   - Once the NameNode service restarts, its status must be displayed in both the vSphere UI and in the status indicator of HAM.
   - The blocked LS operation started in HAM should now continue without failures.

Verify that no fail over is triggered on planned shutdown of the monitor service

This test verifies that if the monitor service is shut down the fail over is not triggered. The NameNode can now be manipulated as part of planned management operations.

1. Start the NameNode VM and SSH to the NameNode VM.
2. Terminate the monitor process.

```
service hmonitor-namenode-monitor stop
```

Note

A **kill -9** command for the monitor is not a graceful shutdown and will trigger fail over.

3. Terminate the NameNode process.

```
service hadoop-namenode stop
```

4. Ensure that you see the following expected results:

   - In the vSphere Management UI, the NameNode VM should be live.
   - The fail over should not be initiated by vSphere.
   - In HAM, the NameNode status area should indicate offline status. The non-blocking operations should now report failure.
   - The SSH connection must not be broken and the VM should be live.

5. Restart the monitor process.

```
service hmonitor-namenode-monitor start
```

6. Restart the NameNode process.

```
service hadoop-namenode start
```

7. The NameNode health should be monitored and failures should trigger fail over.

Verify that the monitor provides a bootstrap period before reporting that the NameNode is not live

This test verifies that the monitoring process includes a bootstrap period.

The bootstrap period ensures that the monitor will immediately not report a failure and trigger a restart. Instead, the monitor provides the service a bootstrap period in which probes are allowed to (initially fail). This bootstrap period is configurable (see: Tuning the bootstrap timeout [17]).

1. Start the NameNode VM and SSH to the NameNode VM.
2. Terminate the monitor process.

```
service hmonitor-namenode-monitor stop
```

3. Terminate the NameNode process.

```
service hadoop-namenode stop
```

4. Restart the monitor process.

```
service hmonitor-namenode-monitor start
```

Verify that no fail over is triggered when the NameNode enters the safe mode

This test verifies that the VM is not restarted if the NameNode enters safe mode. This allows administration operations to be performed on a file system in safemode without having to disable HA services.

1. Start the NameNode VM and SSH to the NameNode VM.
2. Enter safe mode.

```
hadoop dfsadmin -safemode enter
```

3. Ensure that you see the following expected results:
   - In the vSphere UI, the NameNode VM should be live.
   - The SSH session should exist and the VM should be live.
4. Terminate the NameNode process.

```
service hadoop-namenode stop
```

5. The vSphere should identify the NameNode failure and should restart the VM.

Note

This test shows that even in safe mode the fail over is triggered if the NameNode process is terminated. To avoid automatic restart for NameNode after performing safe mode operations, use the **service hmonitor-namenode-monitor restart** to restart the monitor service.

2.6. Administration Guide for Highly Available NameNode

In this section:

2.6.1. NameNode shutdown for planned maintenance

Use the following instructions to ensure controlled shutdown of the NameNode server:

1. Shutdown the NameNode monitoring agent.

```
service stop hmonitor-namenode-monitor
```

2. Shutdown the NameNode process.

```
service stop hadoop-namenode
```

2.6.2. Starting the NameNode

Use the following instructions to start the NameNode:

1. Start NameNode process.

```
service start hadoop-namenode
```

2. Start the NameNode monitoring agent and register it with vSphere.

```
service start hmonitor-namenode-monitor
```

3. Verify that the monitoring process has started.

```
service status hmonitor-namenode-monitor
```

2.6.3. Reconfiguring HDFS

Use the following instructions to reconfigure HDFS:

1. Disable HA on vSphere.
2. Shut down HDFS using the instructions available here.
3. Update the HDFS parameters using HDP. See the instructions here

   Restart HDFS using HDP.See the instructions here
4. Enable HA on vSphere.

2.6.4. Tuning parameters for your environment

In this section:

Tuning parameters for your environment

When the VM starts, the HA Monitor waits for the NameNode to begin responding to file system operations. During this "bootstrap phase", the HA monitor does not report startup failures of NameNode probes to the HA infrastructure. The HA monitor exits the bootstrap phase once all the probes succeed (from that point, the failure of a probe is reported as a service failure).

The time limit of the bootstrap phase can be configured using the **service.monitor.bootstrap.timeout** property:

```
<property>
<name>service.monitor.bootstrap.timeout</name>
<value>120000</value>
<description>
The time in milliseconds for the monitor to wait for the service to bootstrap
 and
become available before it reports a failure to the management infrastructure
</description>
</property>
```

The timeout must be sufficiently long so that the monitored service is able to open its network ports for external interaction. For the NameNode, the web page and IPC port must be open.

The bootstrap time also needs to include the time required for the HDFS journal replay operations. The bootstrap timeout value should be kept high if the filesystem is large and if the secondary NameNode checkpointing time intervals are longer.

Tuning the reporting rate

The internal VM Monitor daemon sends "heartbeat" messages to vSphere to indicate that the VM is alive. Use the following property, to modify the rate at which these heartbeats are sent.

```
<property>
<name>service.monitor.report.interval</name>
<value>7000</value>
<description>
Interval in milliseconds for sending heartbeats to vSphere.
</description>
</property>
```

It is essential that the live VM sends a heartbeats to vSphere at least every thirty seconds.

A smaller reporting interval reduces the risk of missed heartbeats in case an Operating System or Java related process hangs. However, a smaller reporting interval can also have adverse effects - especially if the VM is overloaded. It is therefore strongly recommended to address the root cause of the VM overload. If your VM is overloaded and becomes unresponsive, we recommend that you either add more CPUs and RAM or rebalance VMs across the cluster.

Tuning the probing rate

The Monitor daemon probes the health of the NameNode at a regular interval, and stops sending heartbeats to vSphere when any of the probes fail. Use the following property to change the rate of probes:

```
<property>
<name>service.monitor.probe.interval</name>
<value>11000</value>
<description>
Time in milliseconds between the last probe cycle ending and the new one
 beginning.
The shorter this cycle, the faster failures are detected, but more CPU,
 network,
and server load can be generated.
</description>
</property>
```

The smaller the interval between probes, the faster it becomes to detect and report service failures. Although, this might increase the load on the service and the CPU slightly, but even with a very short probing interval, vSphere will not trigger VM restart for at least thirty seconds after the probe failure.

Tuning for NameNode Garbage Collection

The NameNode process can appear hung during Garbage Collection event. To prevent this from triggering immediate failover, a grace period is provided to the NameNode to resume its operation. You can configure this grace period using the following property:

```
<property>
<name>service.monitor.probe.timeout</name>
<value>60000</value>
<description>
Duration in milliseconds for the probe loop to be blocked, before it is
 considered a liveness failure
</description>
</property>
```

A smaller value will cause the VM (where the hung NameNode process is running) faster, but it increases the risk of incorrectly identifying a long GC-related pause as a hung process. On larger clusters (with longer GC pauses), you can increase the value of this property.

2.7. References

- High Availability for Hadoop using VMWare

3. High Availability for Hadoop Using Red Hat

This section describes using Red Hat configurations to support high availability.

3.1. High Availability for Hadoop Using Red Hat

This document is intended for system administrators for configuring the Hadoop NameNode (NN) service for High Availability and also for configuring clients and other Hadoop services to be resilient while the NameNode fails over.

The Hortonworks Data Platform (HDP) High Availability (HA) kit for Red Hat Enterprise Linux (RHEL) provides enterprise grade high availability of the NameNode Service as part of the HDP cluster.

To learn more, see HDP's Full-Stack HA Architecture.

In this document:

- Use Cases and Fail Over Scenarios
- Typical HDP HA Cluster
- Prerequisites
- Install HDP Hadoop Core Packages
- Deploy HDP HA Configurations
- Configure NameNode HA for RHEL Cluster
- Distribute Cluster Configuration
- Validate Cluster Fail Over

3.2. Use Cases and Fail Over Scenarios

This section provides information on the following:

- Supported use cases
- Supported fail over scenarios

3.2.1. Supported use cases

This solution enables the following Hadoop system administration use cases:

- Planned failure of the NameNode (maintenance tasks like software or hardware upgrade)

- Unplanned failure of the NameNode (hardware or OS failure, software bugs, JVM related issues, etc.)

3.2.2. Supported fail over scenarios

The following failover scenarios are supported:

- NameNode service failure
- NameNode JVM failure
- Hung NameNode daemon or hung operating system
- NameNode operating system failure
- NameNode machine power failure
- Failure of NIC cards on the NameNode machine
- Network failure for the NameNode machine

3.3. Typical HDP HA Cluster

When you configure HDP HA solution using RHEL, your Hadoop cluster comprises the following two components:

- RHEL HA cluster
- Overall Hadoop cluster

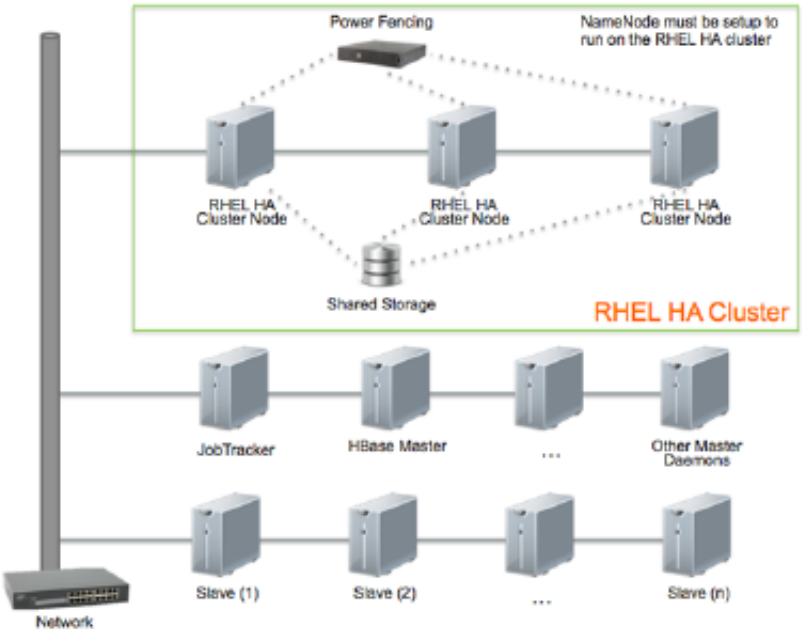
Important

The RHEL HA cluster is a subset of the overall Hadoop cluster.

Typically, the overall Hadoop cluster must include the following types of machines:

- The RHEL HA cluster machines. These machines must host only those master services that require HA (in this case, the NameNode).
- Master machines that run other master services such as JobTracker, HBase master, etc.
- Slave machines that run slave daemons like Datanodes, TaskTracker, RegionServers etc.
- Client machines.
- A network that connects all the machines.

The following illustrates the topology of a sample Hadoop cluster when configured for HDP HA using RHEL:



3.4. Prerequisites

3.4.1. Hardware prerequisites

Ensure that you complete the following hardware prerequisites:

- Shared Storage
- Power Fencing device
- IP fail over with a floating IP
- Hardware requirement for RHEL HA cluster

3.4.1.1. Shared Storage

- Shared storage is required for storing the NameNode metadata. Use a highly available shared storage NFS device.

3.4.1.2. Power Fencing Device

- Ensure that you use a Power fencing device.

Note

Red Hat HA cluster utilizes power fencing to deal with network split-brain events. Fencing guarantees the integrity of NameNode metadata. For more information, see: Fencing Topology.

3.4.1.3. IP Fail Over

- Ensure that an additional static IP is available for the cluster.

- The IP must be a static reserved entry in your network DNS table. This IP will act as the public IP for the NameNode Service.

Note

Red Hat HA clustering utilizes a floating IP address for the NameNode service across the HA cluster. More details on using a floating IP for RHEL are available here.

3.4.1.4. Hardware Requirements for RHEL HA Cluster

- The RHEL HA cluster must have a minimum of two nodes.
- The number of nodes in your HA cluster depends on the number of concurrent node failures you want the HDP platform to withstand. The RHEL HA cluster can be configured to include a maximum of 16 nodes. Choose hardware specs for the RHEL HA Cluster nodes according to the NameNode hardware recommendations available here.

3.4.2. Software prerequisites

Ensure that you complete the following software prerequisites:

- Configure RHEL HA cluster
- Validate configurations for RHEL HA cluster

3.4.2.1. Configure RHEL HA Cluster

Step 1: Complete the prerequisites for High Availability Add-On package for RHEL.

- Use the instructions available here (RHEL v5.x., RHEL v6.x).

Step 2: Install the HA Add-On package for RHEL.

- Use the instructions available here (RHEL v5.x.,RHEL v6.x).

Important

You can use the graphical user interface (GUI) to configure a RHEL v6.x cluster configuration until you specify a Hadoop service configuration (Deploy HDP HA Configurations). You must use the `cluster.conf` file to specify the Hadoop service configuration. Once the Hadoop service configuration is put in place, any changes made via the GUI will break the configuration. You can still use the GUI to manage the HA NameNode service – start, stop, and move the service across cluster machines.

Step 3: Ensure that the following cluster configurations are available on all the machines in your RHEL HA cluster:

- Cluster domain that specifies all the nodes in the RHEL HA cluster. See instructions here (RHEL v5.x., RHEL v6.x.).
- Fail over domain. See instructions here (RHEL v5.x., RHEL v6.x.).

- Power Fencing device. See instructions here (RHEL v5.x., RHEL v6.x.).
- Add cluster service and resources (Floating IP and NFS mount). Ensure that you add the `<service domain>` configurations and to add resources to the service group: See instructions here (RHEL v5.x., RHEL v6.x.).
- When the above are configured, you will have a `cluster.conf` file similar to the following sample configuration. (Note that this sample configuration does not declare a true fencing device because that is specific to the environment. Modify the configuration values to match your infrastructure environment.)

```
<?xml version="1.0"?>
<cluster config_version="8" name="rhel6ha">
<clusternodes>
  <clusternode name="rhel6ha01" nodeid="1">
    <fence>
      <method name="1">
        <device name="BinTrue"/>
      </method>
    </fence>
  </clusternode>
  <clusternode name="rhel6ha02" nodeid="2">
    <fence>
      <method name="1">
        <device name="BinTrue"/>
      </method>
    </fence>
  </clusternode>
</clusternodes>
<cman expected_votes="1" two_node="1"/>
<fencedevices>
  <fencedevice agent="fence_bin_true" name="BinTrue"/>
</fencedevices>
<rm log_level="7">
  <failoverdomains>
    <failoverdomain name="HANameNode" ordered="1" restricted="1">
      <failoverdomainnode name="rhel6ha01" priority="1"/>
      <failoverdomainnode name="rhel6ha02" priority="2"/>
    </failoverdomain>
  </failoverdomains>
   <service domain="HANameNode" name="NameNodeService"
           recovery="relocate">
    <ip address="10.10.10.89" sleeptime="10"/>
     <netfs export="/hdp/nfs" force_unmount="1" fstype="nfs" host="10.10.
10.88" mountpoint="/hdp/hadoop/hdfs/nn" name="HDFS data" options="rw,soft,
nolock"/>
  </service>
</rm>
</cluster>
```

3.4.2.2. Validate Configurations for RHEL HA Cluster

Use the following instructions to validate the configurations for RHEL HA cluster.

Step 1: Validate that the floating IP address is available on the primary machine. (Primary machine is the machine where the NameNode process is currently running).

```
ip addr show eth1
```

If the IP address is available, you should see a message (as shown in the following example). In this example, the IP address is configured at `rheln1.hortonworks.local`:

```
root@rheln1 ~]# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 qlen 1000
link/ether 00:0c:29:cb:ca:76 brd ff:ff:ff:ff:ff:ff
inet 172.16.204.10/24 brd 172.16.204.255 scope global eth1
inet 172.16.204.12/24 scope global secondary eth1
inet6 fe80::20c:29ff:fecb:ca76/64 scope link
valid_lft forever preferred_lft forever
```

Step 2: Validate that the NameNode service starts on the secondary machine.

```
ip addr show eth3
```

Step 3: Validate fail over for the IP address.

- Shut down alternate host machines.
- Ensure that the IP address fails over properly.

3.5. Install HDP Hadoop Core Packages

Install Hadoop Core packages on the entire cluster using the instructions provided here.

Ensure that you complete the following set-up requirements for NameNode machines:

- Use the floating IP for the NameNode network location (`$DFS_NAME_DIR`). Add a DNS entry with a hostname that points to the floating IP address. Use this hostname to specify the NameNode host address.
- Configure the NameNode data directory as a child of the mounted shared storage directory. For example, if you have the NFS directory mounted at `/hdp/hdfs/nn`, then the NameNode data directory should be configured to be at the following location: `/hdp/hdfs/nn/data`.
- Replicate the above set-up of the NameNode machine on all the machines in your RHEL HA cluster. Ensure that each machine in the RHEL HA cluster has identical HDP packages and add-on libraries.

3.6. Deploy HDP HA Configurations

Use the instructions provided in this section to configure Full-Stack HA fail over resiliency for the HDP clients.

Note

Your Hadoop configuration directories are defined during the HDP installation. For details, see: Setting Up Hadoop Configuration.

Step 1: Edit the `$HADOOP_CONF_DIR/hdfs-site.xml` file to add the following properties:

- Enable the HDFS client retry policy.

```
<property>
 <name>dfs.client.retry.policy.enabled</name>
 <value>true</value>
 <description> Enables HDFS client retry in case of NameNode failure.</
description>
</property>
```

- Configure protection for NameNode edit log.

```
<property>
 <name>dfs.namenode.edits.toleration.length</name>
 <value>8192</value>
 <description> Prevents corruption of NameNode edit log.</description>
</property>
```

- Configure safe mode extension time.

```
<property>
 <name>dfs.safemode.extension</name>
 <value>10</value>
 <description> The default value (30 seconds) is applicable for very large
 clusters. For small to large clusters (upto 200 nodes), recommended value
 is 10 seconds.</description>
</property>
```

- Ensure that the allocated DFS blocks persist across multiple fail overs.

```
<property>
 <name>dfs.persist.blocks</name>
 <value>true</value>
 <description>Ensure that the allocated DFS blocks persist across multiple
 fail overs.</description>
</property>
```

- Configure delay for first block report.

```
<property>
 <name>dfs.blockreport.initialDelay</name>
 <value>10</value>
 <description> Delay (in seconds) for first block report.</description>
</property>
```

Step 2: Modify the following property in the `$HADOOP_CONF_DIR/core-site.xml` file:

```
<property>
 <name>fs.checkpoint.period</name>
 <value>3600</value>
 <description> The number of seconds between two periodic checkpoints.</
description>
</property>
```

This will ensure that the checkpoint is performed on an hourly basis.

Step 3: Edit the *$HADOOP_CONF_DIR*/`mapred-site.xml` file to add the following properties:

- Enable the JobTracker's safe mode functionality.

```
<property>
 <name>mapreduce.jt.hdfs.monitor.enable</name>
 <value>true</value>
 <description> Enable the JobTracker to go into safe mode when the NameNode
 is not responding.</description>
</property>
```

- Enable retry for JobTracker clients (when the JobTracker is in safe mode).

```
<property>
 <name>mapreduce.jobclient.retry.policy.enabled</name>
 <value>true</value>
 <description> Enable the MapReduce job client to retry job submission when
 the JobTracker is in safe mode.</description>
</property>
```

- Enable recovery of JobTracker's queue after it is restarted.

```
<property>
 <name>mapred.jobtracker.restart.recover</name>
 <value>true</value>
 <description> Enable the JobTracker to recover its queue after it is
 restarted.</description>
</property>
```

3.7. Configure NameNode HA for RHEL Cluster

Follow the tasks listed below to configure NameNode HA:

- Install NameNode monitoring component
- Configure NameNode service in clustering configuration

3.7.1. Install NameNode monitoring component

Step 1: Stop all the HDP services that are currently running on your cluster.

- Use the instructions provided here.

Step 2: Stop the RHEL cluster NameNodeService service.

- Use the RHEL Cluster administration tools (RHEL v5.x., RHEL v6.x.).

Step 3: Install the NameNode monitoring component on all the nodes in your RHEL HA cluster.

- Ensure that you have set up the HDP repository on the RHEL HA cluster nodes as part of the HDP installation. Use the instructions provided here.
- Install the RPMs.
  - For RHEL/CentOS 5

```
        wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/
1.2.1/hamonitor_redhat/hmonitor-resource-agent-1.1.0.23-1.el5.x86_64.rpm
        wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/
1.2.1/hamonitor_redhat/hmonitor-1.1.0.23-1.el5.x86_64.rpm
yum install hmonitor-1.1.0.23-1.el5.x86_64.rpm
yum install hmonitor-resource-agent-1.1.0.23*.rpm
```

  - For RHEL/CentOS 6

```
        wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/
1.2.1/hamonitor_redhat/hmonitor-resource-agent-1.1.0.23-1.el6.x86_64.rpm
        wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/
1.2.1/hamonitor_redhat/hmonitor-1.1.0.23-1.el6.x86_64.rpm
yum install hmonitor-1.1.0.23-1.el6.x86_64.rpm
yum install hmonitor-resource-agent-1.1.0.23*.rpm
```

3.7.2. Configure NameNode service in clustering configuration

Edit the `/etc/cluster/cluster.conf` file to add the service domain specifications. You can use the following sample configuration. (Note that this sample configuration is for a small cluster and the timeouts for booting, probing, and stopping have been reduced to a minimum.)

```
<service domain="HANameNode" name="NameNodeService" recovery="restart">
  <ip address="10.10.10.89" sleeptime="10"/>
  <netfs export="/hdp/hadoop-nfs" force_unmount="1" fstype="nfs"
         host="10.10.10.88" mountpoint="/hdp/hadoop/hdfs/nn"
         name="HDFS data" options="rw,soft,nolock"/>
  <hadoop
         __independent_subtree="1" __max_restarts="10"
         __restart_expire_time="600" name="NameNode Process"
         daemon="namenode" boottime="10000" probetime="10000"
         stoptime="10000"
         url="http://10.0.0.30:50070/dfshealth.jsp"
         pid="/var/run/hadoop/hdfs/hadoop-hdfs-namenode.pid"
```

```
        path="/"/>
</service>
```

The following table explains the parameters used in the above configuration:

Table 3.1. Parameters for configuring NameNode service in clustering configuration

Name	Description	Mandatory/Optional
daemon	Name of the hadoop service which will be started by `bin/hadoop-daemon`.	Mandatory
url	URL to check for the service web page. This should be on the floating IP address.	Mandatory
pid	Process ID of the NameNode process. (Default: “”)	Optional
path	Path under DFS to probe. (Default: /)	Optional
boottime	Time (in milliseconds) to allow for the service to boot up. This must include any activities that take place before the service web pages and IPC services are reachable. For the NameNode, it must include the time for the edit log to be replayed. (Default: 180000)	Optional
probetime	Time (in milliseconds) to allow for a process to respond to liveness probes. This duration must be longer than the maximum expected GC pause. (Default: 120000)	Optional
stoptime	Time (in milliseconds) to allow for a clean shutdown before forcibly killing a process. (Default: 60000)	Optional

3.8. Distribute Cluster Configuration

Step 1: Ensure that `ricci` is running on the current node and on all other nodes of the cluster.

```
service ricci status
```

This command should return the status message similar to the one shown in the following example:

```
ricci (pid  9524) is running...
```

Step 2: Distribute the configuration to other nodes in the cluster.

- Execute the following command from the node that contains the `cluster.conf` file.
  - For RHEL v5.x:

    ```
    ccs_tool update /etc/cluster/cluster.conf
    ```

  - For RHEL v6.x:

    ```
    ```

```
ccs -i -h host --sync --activate
```

Step 3: Validate the updated configuration files.

- SSH into another node in the cluster and execute the following command:

```
cat /etc/cluster/cluster.conf
```

This command should display the updated contents for the `cluster.conf` file.

3.9. Validate Cluster Fail Over

Use the following tests to validate successful fail over for the NameNode master daemon service:

- Validate NameNode restart on primary machine
- Validate NameNode fail over during soft reboot
- Validate NameNode fail over during hard reboot

3.9.1. Validate NameNode restart on primary machine

This test simulates the event where the NameNode master daemon dies on the primary (current) machine. A successful fail over should restart the NameNode process on the primary machine.

- From the primary machine, perform the following command to shutdown the namenode:

```
su - hdfs
/usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR stop namenode
```

- Once the NameNode is shutdown, the monitor must detect the unavailability of the NameNode and should attempt to restart it. This process will take approximately 30 seconds.

3.9.2. Validate NameNode fail over during soft reboot

This test simulates the event where the NameNode server becomes unavailable due to a planned reboot of the primary NameNode machine. A successful fail over should be able to relocate both the floating IP and the NameNode process to the secondary machine in the cluster. When the primary machine completes the restart, the NameNode process should again be relocated to the primary machine.

- Perform a soft reboot of the server.
- During the shutdown process, the other machine in the cluster will be notified and will begin the migration of the service.

- The service should be available on the secondary machine in the cluster until the primary machine has completed its reboot.

3.9.3. Validate NameNode fail over during hard reboot

This test simulates the event when the NameNode server becomes unavailable due to an unplanned reboot of the primary NameNode machine. A successful fail over should be able to relocate both the floating IP and the NameNode process to the secondary machine in the cluster. When the primary machine completes the restart, the NameNode process should again be relocated to the primary machine.

- Perform a hard shutdown of the primary NameNode machine.
- Once the secondary machine in the cluster detects the outage, it must start the NameNode process on the secondary machine.
- The NameNode service should be available on the secondary machine until the primary machine completes its startup routine.

4. High Availability for Hive Metastore

This document is intended for system administrators who need to configure the Hive Metastore service for High Availability.

To learn more, see HDP's Full-Stack HA Architecture.

4.1. Use Cases and Fail Over Scenarios

This section provides information on the use cases and fail over scenarios for high availability (HA) in the Hive metastore.

Use Cases

The metastore HA solution is designed to handle metastore service failures. Whenever a deployed metastore service goes down, metastore service can remain unavailable for a considerable time until service is brought back up. To avoid such outages, deploy the metastore service in HA mode.

Deployment Scenarios

We recommend deploying the metastore service on multiple boxes concurrently. Each Hive metastore client will read the configuration property `hive.metastore.uris` to get a list of metastore servers with which it can try to communicate.

```
<property>
 <name> hive.metastore.uris </name>
 <value> thrift://$Hive_Metastore_Server_Host_Machine_FQDN </value>
 <description> A comma separated list of metastore uris on which metastore
 service is running </description>
 </property>
```

These metastore servers store their state in a MySQL HA cluster, which should be set up as recommended in the whitepaper *"MySQL Replication for Failover Protection."*

In the case of a secure cluster, each of the metastore servers will additionally need to have the following configuration property in its `hive-site.xml` file.

```
<property>
 <name> hive.cluster.delegation.token.store.class </name>
 <value> org.apache.hadoop.hive.thrift.DBTokenStore </value>
 </property>
```

Fail Over Scenario

A Hive metastore client always uses the first URI to connect with the metastore server. In case the metastore server becomes unreachable, the client will randomly pick up a URI from the list and try connecting with that.

4.2. Software Configuration

Complete the following tasks to configure Hive HA solution:

- Install HDP
- Validate configuration

4.2.1. Install HDP

Use the following instructions to install HDP on your cluster hardware. Ensure that you specify the virtual machine (configured in the previous section) as your NameNode.

1. Download Apache Ambari using the instructions provided here.

Note

Do not start the Ambari server until you have configured the relevant templates as outlined in the following steps.

2. Edit the `<master-install-machine-for-Hive-Metastore>/TODO` file to add the following properties:

   a. Provide the URI for client to contact Metastore server. The following property can have a comma separated list when your cluster has multiple Hive Metastore servers.

```
<property>
 <name> hive.metastore.uris </name>
 <value> thrift://$Hive_Metastore_Server_Host_Machine_FQDN </value>
 <description> URI for client to contact metastore server </description>
</property>
```

   b. Configure Hive cluster delegation token storage class.

```
<property>
 <name> hive.cluster.delegation.token.store.class </name>
 <value> org.apache.hadoop.hive.thrift.DBTokenStore </value>
 </property>
```

3. Complete HDP installation.

   - Continue the Ambari installation process using the instructions provided here.
   - Complete the Ambari installation. Ensure that the installation was successful.

4.2.2. Validate configuration

Test various fail over scenarios to validate your configuration.

5. Upgrade HDP Manually

This document provides instructions on how to upgrade to the latest release of HDP from an earlier HDP release. Use the following instructions to upgrade to the latest release of HDP:

1. Getting Ready to Upgrade
2. Upgrade Hadoop
3. Upgrade ZooKeeper and HBase
4. Upgrade Hive and HCatalog
5. Upgrade Oozie
6. Upgrade WebHCat (Templeton)
7. Upgrade Pig
8. Upgrade Sqoop
9. Upgrade Flume

5.1. Getting Ready to Upgrade

1. Stop all services (including MapReduce) and client applications deployed on HDFS using the instructions provided here.
2. Run the `fsck` command as instructed below and fix any errors. (The resulting file will contain complete block map of the file system.)

   ```
   hadoop fsck / -files -blocks -locations > dfs-old-fsck-1.log
   ```

3. Use the following instructions to compare the status before and after the upgrade:

   Note

   The following commands must be executed by the user running the HDFS service (by default, the user is `hdfs`).

   a. Capture the complete namespace of the file system. (The following command does a recursive listing of the root file system.)

   ```
   su hdfs
   hadoop dfs -lsr / > dfs-old-lsr-1.log
   ```

   b. Run report command to create a list of DataNodes in the cluster.

   ```
   su hdfs
   hadoop dfsadmin -report > dfs-old-report-1.log
   ```

   c. Optionally, copy all or unrecoverable data stored in DFS to a local file system or to a backup instance of DFS.

d. Optionally, repeat the steps 3 (a) through 3 (c) and compare the results with the previous run to ensure the state of the file system remained unchanged.

4. As HDFS user, execute the following command to save namespace:

```
hadoop dfsadmin -saveNamespace
```

5. Copy the following checkpoint files into a backup directory:

   - dfs.name.dir/edits
   - dfs.name.dir/image/fsimage

6. Stop the HDFS. Ensure all the HDP services in the cluster are completely stopped at this point.

7. If upgrading Hive, ensure that you back up the Hive database.

8. Configure the local repositories.

   The standard HDP install fetches the software from a remote yum repository over the Internet. To use this option, you must set up access to the remote repository and have an available Internet connection for each of your hosts.

Note

If your cluster does not have access to the Internet, or you are creating a large cluster and you want to conserve bandwidth, you can instead provide a local copy of the HDP repository that your hosts can access. For more information, see Deployment Strategies for Data Centers with Firewalls., a separate document in this set.

   a. For each node in your cluster, download the yum repo configuration file `hdp.repo`. From a terminal window, type:

      - For RHEL and CentOS 5

```
wget  http://public-repo-1.hortonworks.com/HDP/centos5/1.x/GA/hdp.repo
 -O /etc/yum.repos.d/hdp.repo
```

      - For RHEL and CentOS 6

```
wget  http://public-repo-1.hortonworks.com/HDP/centos6/1.x/GA/hdp.repo
 -O /etc/yum.repos.d/hdp.repo
```

      - For SLES 11

```
wget  http://public-repo-1.hortonworks.com/HDP/suse11/1.x/GA/hdp.repo -
O /etc/zypp/repos.d/hdp.repo
```

   b. Confirm the HDP repository is configured by checking the repo list.

      - For RHEL/CentOS:

```
yum repolist
```

- For SLES:

```
zypper repos
```

5.2. Upgrade Hadoop

1. On all nodes:
   - For RHEL/CentOS:

```
yum upgrade hadoop\*
```

   - For SLES:

```
zypper update hadoop\*
```

2. Start HDFS.
   a. Start NameNode. On the NameNode host machine, execute the following command:

```
sudo su -l hdfs -c "/usr/lib/hadoop/bin/hadoop-daemon.sh start namenode"
```

   b. Start Secondary NameNode. On the Secondary NameNode host machine, execute the following command:

```
sudo su -l hdfs -c "/usr/lib/hadoop/bin/hadoop-daemon.sh start
 secondarynamenode"
```

   c. Start DataNodes. On all the DataNodes, execute the following command:

```
sudo su -l hdfs -c \"/usr/lib/hadoop/bin/hadoop-daemon.sh start datanode\
"
```

   d. Execute the following on the NameNode machine:

```
hadoop dfsadmin -safemode wait
```

3. Start MapReduce.
   a. Start JobTracker. On the JobTracker host machine, execute the following command:

```
sudo su -l mapred -c "/usr/lib/hadoop/bin/hadoop-daemon.sh start
 jobtracker"
```

   b. Start JobHistory Server. On the JobHistory Server host machine, execute the following command:

```
sudo su -l mapred -c "/usr/lib/hadoop/bin/hadoop-daemon.sh start
 historyserver"
```

   c. Start all TaskTrackers. On all the TaskTrackers, execute the following command:

```
sudo su -l mapred -c \"/usr/lib/hadoop/bin/hadoop-daemon.sh start
 tasktracker\"
```

5.3. Upgrade ZooKeeper and HBase

1. Execute the following command on all the ZooKeeper nodes:

   - For RHEL/CentOS:

     ```
     yum upgrade zookeeper
     ```

   - For SLES:

     ```
     zypper update zookeeper
     ```

2. Start ZooKeeper. On all the ZooKeeper host machines, execute the following command:

   ```
   sudo su -l zookeeper -c \"source /etc/zookeeper/conf/zookeeper-env.sh;
   export ZOOCFGDIR=/etc/zookeeper/conf; /usr/lib/zookeeper/bin/zkServer.sh
    start >> $ZOOKEEPER_LOG_DIR/zoo.out\"
   ```

   where *$ZOOKEEPER_LOG_DIR* is the directory where ZooKeeper server logs are stored. For example, `/var/log/zookeeper`.

3. Execute the following commands on the HBase Master and the HBase slave nodes (RegionServers):

   - For RHEL/CentOS:

     ```
     yum upgrade hbase
     ```

   - For SLES:

     ```
     zypper update hbase
     ```

4. Start HBase.

   a. Start HBase Master. On the HBase Master host machine, execute the following command:

      ```
      sudo su -l hbase -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
      hbase/conf start master"
      ```

   b. Start all RegionServers. On all the RegionServers, execute the following command:

      ```
      sudo su -l hbase -c \"/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
      hbase/conf start regionserver\"
      ```

5.4. Upgrade Hive and HCatalog

1. Upgrade Hive and HCatalog. On the Hive and HCatalog host machines, execute the following command:s

   - For RHEL/CentOS:

     ```
     yum upgrade hive hcatalog
     ```

   - For SLES:

     ```
     zypper update hive hcatalog
     ```

2. Start Hive. On the Hive Metastore host machine, execute the following command:

   ```
   sudo su -l hive -c "nohup hive --service metastore > $HIVE_LOG_DIR/hive.out
   2> $HIVE_LOG_DIR/hive.log &"
   ```

   where *$HIVE_LOG_DIR* is the directory where Hive server logs are stored (example: `/var/log/hive`).

3. Start Hive Server2. On the Hive Server2 host machine, execute the following command:

   ```
   sudo su -l hive -c "nohup /usr/lib/hive/bin/hiveserver2 -hiveconf hive.
   metastore.uris=\" \" > $HIVE_LOG_DIR/hiveserver2.out 2> $HIVE_LOG_DIR/
   hiveserver2.log &"
   ```

   where *$HIVE_LOG_DIR* is the directory where Hive server logs are stored (example: `/var/log/hive`).

5.5. Upgrade Oozie

1. Execute the following command on the Oozie server and client machines:

   - For RHEL/CentOS:

     ```
     yum upgrade oozie
     ```

   - For SLES:

     ```
     zypper update oozie
     ```

2. Start Oozie.

   ```
   sudo su -l oozie -c "cd $OOZIE_LOG_DIR/log; /usr/lib/oozie/bin/oozie-start.
   sh"
   ```

   where *$OOZIE_LOG_DIR* is the directory where Oozie log files are stored (for example: `/var/log/oozie`).

3. Validate Oozie.

```
oozie admin -oozie http://$fully.qualified.domain.
name_for_Oozie_hostmachine:11000/oozie -status
```

You should see the following output:

```
System mode: NORMAL
```

5.6. Upgrade WebHCat (Templeton)

1. Remove old Templeton packages. On the Templeton host machine, execute the following commands:
   - For RHEL/CentOS:

```
yum remove templeton\*
```

   - For SLES:

```
zypper remove templeton\*
```

2. Install WebHCat.
   - For RHEL/CentOS:

```
yum install webhcat-tar-hive webhcat-tar-pig
```

   - For SLES:

```
zypper install webhcat-tar-hive webhcat-tar-pig
```

   Also see the instructions on manually deploying WebHCat instance provided here.

3. Start WebHCat. On the WebHCat host machine, execute the following command:

```
sudo su -l hcat -c "/usr/lib/hcatalog/sbin/webhcat_server.sh start"
```

4. Smoke test WebHCat. On the WebHCat host machine, execute the following command:

```
http://$WEBHCAT_HOST_MACHINE:50111/templeton/v1/status
```

5. Remove shared libraries from old Templeton installation. On the WebHCat host machine, execute the following command:

```
sudo su -l hdfs -c "hadoop dfs -rmr -skipTrash /apps/templeton"
rm -rf /usr/share/HDP-templeton
```

5.7. Upgrade Pig

1. On all the Pig clients, execute the following command:
   - For RHEL/CentOS:

```
yum upgrade pig
```

- For SLES:

```
zypper update pig
```

5.8. Upgrade Sqoop

Upgrade Sqoop. On the Sqoop host machine, execute the following command:

- For RHEL/CentOS:

```
yum upgrade sqoop
```

- For SLES:

```
zypper update sqoop
```

5.9. Upgrade Flume

Upgrade Flume. On the Flume host machine, execute the following command:

- For RHEL/CentOS:

```
yum upgrade flume
```

- For SLES:

```
zypper update flume
```

6. Manually Add Slave Nodes to HDP Cluster

Use the following instructions to manually add slave nodes to your HDP cluster:

1. Prerequisites
2. Add DataNodes or TaskTrackers
3. Add HBase RegionServer
4. Optional - Configure Monitoring Using Ganglia
5. Optional - Configure Cluster Alerting Using Nagios

6.1. Prerequisites

Ensure that the new slave nodes meet the following prerequisites:

- The following operating systems are supported:
  - 64-bit Red Hat Enterprise Linux (RHEL) 5 or 6
  - 64-bit CentOS 5 or 6
  - 64-bit SUSE Linux Enterprise Server (SLES) 11, SP1
- On each of your hosts:
  - yum (RHEL)
  - zypper (SLES)
  - rpm
  - scp
  - curl
  - wget
  - unzip
  - tar
  - pdsh
- Ensure that all the ports listed here are available to the Installer.
- To install Hive metastore or to use external database for Oozie metastore, ensure that you deploy either a MySQL or an Oracle database in your cluster. For instructions, see here.

- Your system must have the correct JDK installed on all the nodes of the cluster. HDP requires Oracle JDK 1.6 update 31. For more information, see Install the Java Development Kit.

6.2. Add DataNodes or TaskTrackers

Use the following instructions to manually add a DataNode or a TaskTracker hosts:

1. On each of the newly added slave nodes, add the HDP repository to yum:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/repos/centos6/hdp.repo -O
/etc/yum.repos.d/hdp.repo
yum clean all
```

2. On each of the newly added slave nodes, install HDFS and MapReduce.

   - On RHEL and CentOS:

```
yum install hadoop hadoop-libhdfs hadoop-native
yum install hadoop-pipes hadoop-sbin openssl
```

   - On SLES:

```
zypper install hadoop hadoop-libhdfs hadoop-native
zypper install hadoop-pipes hadoop-sbin openssl
```

3. On each of the newly added slave nodes, install Snappy compression/decompression library:

   a. Check if Snappy is already installed:

```
rpm-qa | grep snappy
```

   b. Install Snappy on the new nodes:

      - For RHEL/CentOS:

```
yum install snappy snappy-devel
```

      - For SLES:

```
zypper install snappy snappy-devel
```

```
ln -sf /usr/lib64/libsnappy.so
/usr/lib/hadoop/lib/native/Linux-amd64-64/.
```

4. Optional - Install the LZO compression library.

   - On RHEL and CentOS:

```
yum install lzo-devel hadoop-lzo-native
```

   - On SLES:

```
zypper install lzo-devel hadoop-lzo-native
```

5. Copy the Hadoop configurations to the newly added slave nodes and set appropriate permissions.

- **Option I:** Copy Hadoop config files from an existing slave node.

  a. On an existing slave node, make a copy of the current configurations:

  ```
  tar zcvf hadoop_conf.tgz /etc/hadoop/conf
  ```

  b. Copy this file to each of the new nodes:

  ```
  rm -rf /etc/hadoop/conf
  cd /
  tar zxvf $location_of_copied_conf_tar_file/hadoop_conf.tgz
  chmod -R 755 /etc/hadoop/conf
  ```

- **Option II:** Manually add Hadoop configuration files.

  a. Download core Hadoop configuration files from here and extract the files under `configuration_files -> core_hadoop` directory to a temporary location.

  b. In the temporary directory, locate the following files and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

Table 6.1. core-site.xml

Property	Example	Description
fs.default.name	`hdfs://{namenode.full.host-name}:8020`	Enter your NameNode hostname
fs.checkpoint.dir	`/grid/hadoop/hdfs/snn`	A comma separated list of paths. Use the list of directories from `$FS_CHECKPOINT_DIR.` .

Table 6.2. hdfs-site.xml

Property	Example	Description
dfs.name.dir	`/grid/hadoop/hdfs/nn,/grid1/hadoop/hdfs/nn`	Comma separated list of paths. Use the list of directories from `$DFS_NAME_DIR`
dfs.data.dir	`/grid/hadoop/hdfs/dn,grid1/hadoop/hdfs/dn`	Comma separated list of paths. Use the list of directories from `$DFS_DATA_DIR`
dfs.http.address	`{namenode.full.hostname}:50070`	Enter your NameNode hostname for http access
dfs.secondary.http.address	`{secondary.namenode.full.host-name}:50090`	Enter your SecondaryNameNode hostname
dfs.https.address	`{namenode.full.hostname}:50470`	Enter your NameNode hostname for https access.

Table 6.3. mapred-site.xml

Property	Example	Description
mapred.job.tracker	`{jobtracker.full.host-name}:50300`	Enter your JobTracker hostname
mapred.job.tracker.http.address	`{jobtracker.full.host-name}:50030`	Enter your JobTracker hostname
mapred.local.dir	`/grid/hadoop/mapred,/grid1/hadoop/mapred`	Comma separated list of paths. Use the list of directories from `$MAPREDUCE_LOCAL_DIR`

Property	Example	Description
mapreduce.tasktracker.group	hadoop	Enter your group. Use the value of $HADOOP_GROUP
mapreduce.history.server.http.address	{jobtracker.full.hostname}:51111	Enter your JobTracker hostname

Table 6.4. taskcontroller.cfg

Property	Example	Description
mapred.local.dir	/grid/hadoop/mapred,/grid1/hadoop/mapred	Comma separated list of paths. Use the list of directories from *$MAPREDUCE_LOCAL_DIR*

c. Create the config directory on all hosts in your cluster, copy in all the configuration files, and set permissions.

```
rm -r $HADOOP_CONF_DIR
mkdir -p $HADOOP_CONF_DIR
```

```
 <copy the all the config files to $HADOOP_CONF_DIR>
```

```
chmod a+x $HADOOP_CONF_DIR/
chown -R $HDFS_USER:$HADOOP_GROUP $HADOOP_CONF_DIR/../
chmod -R 755 $HADOOP_CONF_DIR/../
```

where:

- *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

6. On each of the newly added slave nodes, start HDFS:

```
su -hdfs
/usr/lib/hadoop/bin/hadoop-daemon.sh --config
$HADOOP_CONF_DIR start datanode
```

7. On each of the newly added slave nodes, start MapReduce:

```
su -mapred
/usr/lib/hadoop/bin/hadoop-daemon.sh --config
$HADOOP_CONF_DIR start tasktracker
```

8. Add new slave nodes.

- To add a new NameNode slave (DataNode):

a. On the NameNode host machine, edit the `/etc/hadoop/conf/dfs.include` file and add the list of slave nodes' hostnames (separated by newline character).

Important

Ensure that you create a new `dfs.include` file, if the NameNode host machine does not have an existing copy of this file.

b. On the NameNode host machine, execute the following command:

```
su – hdfs –c “hadoop dfsadmin –refreshNodes”
```

- To add a new JobTracker slave (TaskTracker):

a. One the JobTracker host machine, edit the `/etc/hadoop/conf/mapred.include` file and add the list of slave nodes' hostnames (separated by newline character).

Important

Ensure that you create a new `mapred.include` file, if the JobTracker host machine does not have an existing copy of this file.

b. On the JobTracker host machine, execute the following command:

```
su – mapred –c “hadoop mradmin –refreshNodes”
```

9. Optional - Enable monitoring on the newly added slave nodes using the instructions provided here.

10. Optional - Enable cluster alerting on the newly added slave nodes using the instructions provided here.

6.3. Add HBase RegionServer

Use the following instructions to manually add HBase RegionServer hosts:

1. On each of the newly added slave nodes, install HBase and ZooKeeper.

   - For RHEL/CentOS:

   ```
   yum install zookeeper hbase
   ```

   - For SLES:

   ```
   zypper install zookeeper hbase
   ```

2. On each of the newly added slave nodes, add the HDP repository to yum:

```
wget -nv http://public-repo-1.hortonworks.com/HDP-1.2.2/repos/centos6/hdp.
repo -O
/etc/yum.repos.d/hdp.repo
yum clean all
```

3. Copy the HBase configurations to the newly added slave nodes and set appropriate permissions.

   - **Option I:** Copy HBase config files from an existing slave node.

   a. On any existing slave node, make a copy of the current configurations:

   ```
   tar zcvf hbase_conf.tgz /etc/hbase/conf
   tar zcvf zookeeper_conf.tgz /etc/zookeeper/conf
   ```

b. Copy this file to each of the new nodes:

```
rm -rf /etc/hbase/conf
mkdir -p /etc/hbase/conf
cd /
tar zxvf $location_of_copied_conf_tar_file/hbase_conf.tgz
chmod -R 755 /etc/hbase/conf
```

```
rm -rf /etc/zookeeper/conf
mkdir -p /etc/zookeeper/conf
cd /
tar zxvf $location_of_copied_conf_tar_file/zookeeper_conf.tgz
chmod -R 755 /etc/zookeeper/conf
```

- **Option II:** Manually add Hadoop configuration files.

a. Download the HBase/ZooKeeper config files from here and extract these files under `configuration_files -> hbase` and `configuration_files -> zookeeper` directories to two temporary locations.

b. Modify the configuration files:

Table 6.5. zoo.cfg

Variable	Example	Description
server.1	*$zookeeper.server1.full.hostname* :2888:3888	Enter the 1st ZooKeeper hostname
server.2	*$zookeeper.server1.full.hostname* :2888:3888	Enter the 2nd ZooKeeper hostname
server.3	*$zookeeper.server3.full.hostname* :2888:3888	Enter the 3rd ZooKeeper hostname

Table 6.6. hbase-site.xml

Variable	Example	Description
hbase.rootdir	hdfs://*$namenode.full.hostname* :8020/apps/hbase/data	Enter the HBase NameNode server
hbase.master.info.bindAddress	*$hbase master.full.hostname*	Enter the HBase Master server hostname
hbase.zookeeper.quorum	server1.full.hostname, server2.full.hostname, server3.full.hostname	Comma separated list of Zookeeper servers (match to what is specified in zoo.cfg but without port numbers)

4. On all the new slave nodes create the config directory, copy all the config files, and set the permissions:

```
rm -r $HBASE_CONF_DIR ;
mkdir -p $HBASE_CONF_DIR ;
```

```
copy all the config files to $HBASE_CONF_DIR
```

```
chmod a+x $HBASE_CONF_DIR/;
chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_CONF_DIR/../  ;
chmod -R 755 $HBASE_CONF_DIR/../
```

```
rm -r $ZOOKEEPER_CONF_DIR ;
mkdir -p $ZOOKEEPER_CONF_DIR ;
```

```
copy all the config files to $ZOOKEEPER_CONF_DIR
```

```
chmod a+x $ZOOKEEPER_CONF_DIR/;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_CONF_DIR/../  ;
chmod -R 755 $ZOOKEEPER_CONF_DIR/../
```

where:

- *$HBASE_CONF_DIR* is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`.
- *$HBASE_USER* is the user owning the HBase services. For example, `hbase`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.
- *$ZOOKEEPER_CONF_DIR* is the directory to store the ZooKeeper configuration files. For example, `/etc/zookeeper/conf`
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.

5. Start HBase RegionServer node:

```
<login as $HBASE_USER>
/usr/lib/hbase/bin/hbase-daemon.sh --config $HBASE_CONF_DIR start
 regionserver
```

6. On the HBase Master host machine, edit the `/usr/lib/hbase/conf` file and add the list of slave nodes' hostnames. The hostnames must be separated by a newline character.
7. **Optional:** Enable monitoring on the newly added slave nodes using the instructions provided here.
8. **Optional:** Enable cluster alerting on the newly added slave nodes using the instructions provided here.

6.4. Optional - Configure Monitoring Using Ganglia

For each of the newly added slave node, complete the following instructions in order to use Ganglia for monitoring and metrics collection:

1. On each new host, install the Ganglia gmond:
   - On RHEL and CentOS:

     ```
     yum install ganglia-gmond-3.2.0-99
     ```

   - On SLES:

     ```
     zypper install ganglia-gmond-3.2.0-99
     ```

2. Copy the HDP Ganglia scripts from any existing slave node to the newly added node:

```
mkdir -p /usr/libexec/hdp/ganglia
scp root@$FQDN_of_any_existing_node:/usr/libexec/hdp/ganglia/*
/usr/libexec/hdp/ganglia
```

3. Configure the gmond emitter:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPSlaves
```

4. Edit the `/etc/ganglia/hdp/HDPSlaves/conf.d/gmond.slave.conf`to validate the slave configuration files.

   Ensure that `host` points to the Ganglia server master node as shown below:

```
host = $Ganglia_server_hostname
```

5. Copy Ganglia service start file from any existing slave node to each of the newly added node:

```
scp root@$existing_node:/etc/init.d/hdp-gmond /etc/init.d
/etc/init.d/hdp-gmond start
```

   This step will start Ganglia monitoring for your Hadoop cluster.

6.5. Optional - Configure Cluster Alerting Using Nagios

For each of the newly added slave node, complete the following instructions in order to use Nagios for cluster alerting:

1. On the Nagios server, add the host definition to the `/etc/nagios/objects/hadoop-hosts.cfg` file:

2. On the Nagios server, edit the `/etc/nagios/objects/hadoop-hostsgroups.cfg` file.

   Add hostnames of the newly added slave nodes to members list under the `slaves` host group.

3. Restart the Nagios server:

```
service nagios start
```

7. Decommission Slave Nodes

Hadoop provides the decommission feature to retire a set of existing slave nodes (DataNodes, TaskTrackers, or HBase RegionServers) in order to prevent data loss.

Slaves nodes are frequently decommissioned for maintainance. As a Hadoop administrator, you will decommission the slave nodes periodically in order to either reduce the cluster size or to gracefully remove dying nodes.

Use the following sections to decommission slave nodes in your cluster:

- Prerequisites
- Decommission DataNodes or TaskTrackers
- Decommission HBase RegionServers

7.1. Prerequisites

- Ensure that the following property is defined in your `hdfs-site.xml` file.

```
<property>
   <name>dfs.hosts.exclude</name>
   <value>$HADOOP_CONF_DIR/excludes</value>
   <final>true</final>
</property>
```

where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

- Ensure that the following property is defined in your `mapred-site.xml` file.

```
<property>
   <name>mapred.hosts.exclude </name>
   <value>$HADOOP_CONF_DIR/excludes</value>
   <final>true</final>
</property>
```

where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

7.2. Decommission DataNodes or TaskTrackers

Nodes normally run both a DataNode and a TaskTracker, and both are typically commissioned or decommissioned together.

With the replication level set to three, HDFS is resilient to individual DataNodes failures. However, there is a high chance of data loss when you terminate DataNodes without decommissioning them first. Nodes must be decommissioned on a schedule that permits replication of blocks being decommissioned.

On the other hand, if a TaskTracker is shutdown, the JobTracker will schedule the tasks on other TaskTrackers. However, decommissioning a TaskTracker is required especially in situations where you want that TaskTracker to stop to accepting new tasks or when the tasks take time to execute but you still want to be agile in your cluster management.

7.2.1. Decommission DataNodes

Use the following instructions to decommission DataNodes in your cluster:

1. On the NameNode host machine, edit the *$HADOOP_CONF_DIR*/dfs.exclude file and add the list of DataNodes hostnames (separated by newline character).

   where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, /etc/hadoop/conf.

2. Update the NameNode with the new set of permitted DataNodes. On the NameNode host machine, execute the following command:

   ```
   su - $HDFS_USER
   % hadoop dfsadmin –refreshNodes
   ```

   where *$HDFS_USER* is the user owning the HDFS services. For example, hdfs.

3. Open the NameNode web UI (http://*$NameNode_FQDN*:50070) and navigate to **Decommissioning Nodes** page.

   Check whether the admin state has changed to **Decommission In Progress** for the DataNodes being decommissioned.

4. When all the DataNodes report their state as **Decommissioned** then all the blocks have been replicated. Shut down the decommissioned nodes.

5. On the NameNode host machine, remove the decommissioned nodes from the *$HADOOP_CONF_DIR*/dfs.include file and execute the following command:

   ```
   su - $HDFS_USER
   % hadoop dfsadmin –refreshNodes
   ```

7.2.2. Decommission TaskTrackers

Use the following instructions to decommission TaskTrackers in your cluster:

1. On the NameNode host machine, edit the *$HADOOP_CONF_DIR*/mapred.exclude file and add the list of TaskTrackers hostnames (separated by newline character).

   where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, /etc/hadoop/conf.

2. Update the JobTracker with the new set of permitted TaskTrackers. On the JobTracker host machine, execute the following command:

   ```
   su - $MAPRED_USER
   % hadoop mradmin –refreshNodes
   ```

   where *$MAPRED_USER* is the user owning the MapReduce services. For example, mapred.

7.3. Decommission HBase RegionServers

Use the following instructions to decommission HBase RegionServers in your cluster:

1. Decommission RegionServers.

   The preferred method of decommissioning RegionServers is to use the `graceful_stop.sh` script (Option I). This option gradually unloads Regions from the RegionServer, allowing the node to be terminated without impacting data availability. You can also terminate the RegionServer without first unloading its Regions (Option II). This will result in a short window of data unavailability as HBase's natural data recovery operations execute.

   - **Option I: Perform graceful stop**

     You can also use the following command to gracefully decommission a loaded RegionServer.

     Execute the following command from any host machine with HBase configuration installed:

     ```
     su - $HBASE_USER
     /usr/lib/hbase/bin/graceful_stop.sh $RegionServer.Hostname
     ```

     where *$HBASE_USER* is the user owning the HBase Services. For example, `hbase`.

     Important

     The value of *`$RegionServer.Hostname`* argument must match the hostname that HBase uses to identify RegionServers.

     To find the hostname for a particular RegionServer, go to the HBase web UI and check the list of RegionServers in the HBase master UI. Typically, HBase Master uses hostnames but occassionally it can be the FQDN of a RegionServer.

   - **Option II: Use `hbase-daemon.sh`**

     Execute the following command on the RegionServer that you want to decommission:

     Important

     It is important to execute the `hbase-daemon.sh` script on the RegionServer that you want to decommission.

     ```
     su - $HBASE_USER
     /usr/lib/hbase/bin/hbase-daemon.sh stop regionserver
     ```

     where *$HBASE_USER* is the user owning the HBase Services. For example, `hbase`.

     Note that Option II causes the RegionServer to close all the regions before shutting down.

2. Enable the load balancer.

If you used the `graceful_stop.sh` script earlier, you may need to re-enable the Region Balancer. Do so using the `balance_switch` command from the shell. Pass the command `true` to enable the balancer, `false` to disable it. The command's return value is the state of the balancer **before** running the command. If `graceful_stop.sh` disabled the balancer earlier, enable it again like this:

```
su - $HBASE_USER
```

```
hbase shell
```

```
hbase(main):001:0> balance_switch true
false
0 row(s) in 0.3590 seconds
```

where *$HBASE_USER* is the user owning the HBase services. For example, `hbase`