Hortonworks Data Platform

Installing HDP Manually

(Aug 30, 2013)

docs.hortonworks.com

Hortonworks Data Platform : Installing HDP Manually

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.

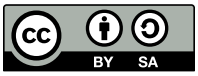


Table of Contents

List of Tables

1. Getting Ready to Install

This section describes the information and materials you need to get ready to install the Hortonworks Data Platform (HDP) manually. Use the following instructions before you deploy Hadoop cluster using HDP:

1. Understand the basics
2. Meet minimum system requirements
3. Configure the remote repositories
4. Decide on deployment type
5. Collect information
6. Prepare the environment
7. Optional - Install MySQL
8. Download companion files
9. Define environment parameters
10. Optional - Create system users and groups

1.1. Understand the Basics

The Hortonworks Data Platform consists of three layers.

- **Core Hadoop**:The basic components of Apache Hadoop.
  - **Hadoop Distributed File System (HDFS)**: A special purpose file system that is designed to work with the MapReduce engine. It provides high-throughput access to data in a highly distributed environment.
  - **Apache Hadoop YARN**: YARN is a general-purpose, distributed, application management framework that supersedes the classic Apache Hadoop MapReduce framework for processing data in Hadoop clusters. The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker i.e. resource management and job scheduling/monitoring, into separate daemons: a global **ResourceManager** and per-application **ApplicationMaster** (AM). The ResourceManager and per-node slave, the **NodeManager** (NM), form the new, and generic, system for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is, in effect, a framework specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks.
  - **MapReduce**: A framework for performing high volume distributed data processing using the MapReduce programming paradigm.

- **Essential Hadoop**A set of Apache components designed to ease working with Core Hadoop.
  - **Apache Pig**: A platform for creating higher level data flow programs that can be compiled into sequences of MapReduce programs, using Pig Latin, the platform's native language.
  - **Apache Hive**: A tool for creating higher level SQL-like queries using HiveQL, the tool's native language, that can be compiled into sequences of MapReduce programs.
  - **Apache HCatalog**: A metadata abstraction layer that insulates users and scripts from how and where data is physically stored.
  - **Apache HBase**: A distributed, column-oriented database that provides the ability to access and manipulate data randomly in the context of the large blocks that make up HDFS.
  - **Apache ZooKeeper**:A centralized tool for providing services to highly distributed systems. ZooKeeper is necessary for HBase installations.
- **Supporting Components**A set of Apache components designed to ease working with Core Hadoop.
  - **Apache Oozie**:A server based workflow engine optimized for running workflows that execute Hadoop jobs.
  - **Apache Sqoop**: A component that provides a mechanism for moving data between HDFS and external structured datastores. Can be integrated with Oozie workflows.

You must always install Core Hadoop, but you can select the components from the other layers based on your needs. For more information on the structure of the HDP, see Understanding Hadoop Ecosystem.

1.2. Meet Minimum System Requirements

To run the Hortonworks Data Platform, your system must meet minimum requirements.

- Hardware Recommendations
- Operating System Requirements
- Software Requirements
- Database Requirements
- JDK Recommendations

1.2.1. Hardware Recommendations

Although there is no single hardware requirement for installing HDP, there are some basic guidelines. You can see sample setups here: Suggested Hardware for a Typical Hadoop Cluster.

1.2.2. Operating Systems Requirements

The following operating systems are supported for Beta:

- 64-bit Red Hat Enterprise Linux (RHEL) 6
- 64-bit CentOS 6

1.2.3. Software Requirements

On each of your hosts:

- yum [for RHEL or CentOS]
- rpm
- scp
- curl
- wget
- unzip
- tar
- pdsh

1.2.4. Database Requirements

By default, Hive and Oozie use Derby database for its metastore. To use external database for Hive and Oozie metastore, ensure that a MySQL database is deployed and available.

- You can choose to use a current instance of MySQL or install a new instance for its use. For more information, see Install MySQL (Optional).
- Ensure that your database administrator creates the following databases and users.
  - For Hive, ensure that your database administrator creates `hive_dbname`, `hive_dbuser`, and `hive_dbpasswd`.
  - For Oozie, ensure that your database administrator creates `oozie_dbname`, `oozie_dbuser`, and `oozie_dbpasswd`.

Note

For instructions on creating users for MySQL, see here.

1.2.5. JDK Requirements

Your system must have the correct JDK installed on all the nodes of the cluster. HDP requires Oracle JDK 1.6 update 31.

Use the following instructions to manually install JDK 1.6 update 31:

1. Check the version. From a terminal window, type:

```
java -version
```

2. (Optional) Uninstall the Java package if the JDK version is less than v1.6 update 31.

```
rpm -qa | grep java
yum remove {java-1.*}
```

3. (Optional) Verify that the default Java package is uninstalled.

```
which java
```

4. Download the Oracle 64-bit JDK (jdk-6u31-linux-x64.bin) from the Oracle download site. From your browser window, go to `http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u31-oth-JPR`.

   Accept the license agreement and download `jdk-6u31-linux-x64.bin` to a temporary directory (**$JDK_download_directory**).

5. Change directory to the location where you downloaded the JDK and run the install.

```
mkdir /usr/jdk1.6.0_31
cd /usr/jdk1.6.0_31
chmod u+x $JDK_download_directory/jdk-6u31-linux-x64.bin
./$JDK_download_directory/jdk-6u31-linux-x64.bin
```

6. Create symbolic links (symlinks) to the JDK.

```
mkdir /usr/java
ln -s /usr/jdk1.6.0_31/jdk1.6.0_31 /usr/java/default
ln -s /usr/java/default/bin/java /usr/bin/java
```

7. Set up your environment to define JAVA_HOME to put the Java Virtual Machine and the Java compiler on your path.

```
export JAVA_HOME=/usr/java/default
export PATH=$JAVA_HOME/bin:$PATH
```

8. Verify if Java is installed in your environment. Execute the following from the command line console:

```
java -version
```

   You should see the following output:

```
java version "1.6.0_31"
Java(TM) SE Runtime Environment (build 1.6.0_31-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
```

1.2.6. Virtualization and Cloud Platforms

HDP is certified and supported when running on virtual or cloud platforms (for example, VMware vSphere or Amazon Web Services EC2) as long as the respective guest operating

system (OS) is supported by HDP and any issues detected on these platforms are reproducible on the same supported OS installed on bare metal.

See Operating Systems Requirements for the list of supported operating systems for HDP.

1.3. Configure the Remote 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.

1. Download the yum repo configuration file `hdp.repo`. On your local mirror server, execute the following command:

   - For RHEL/CentOS 5: [Not supported for Beta]

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.0.
     5.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - For RHEL/CentOS 6:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.0.
     5.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

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

   ```
   yum repolist
   ```

   You should see something like this. Ensure that you have HDP-2.0.5.0 directory:

   ```
   Loaded plugins: fastestmirror, security
   Loading mirror speeds from cached hostfile
   * base: mirrors.cat.pdx.edu
   * extras: linux.mirrors.es.net
   * updates: mirrors.usc.edu
   repo id repo name                                              status
   HDP-2.0.5.0 Hortonworks Data Platform Version - HDP-2.0.5.0 enabled: 53
   ```

1.4. Decide on Deployment Type

While it is possible to deploy all of HDP on a single host, this is appropriate only for initial evaluation. In general you should use at least three hosts: one master host and two slaves.

1.5. Collect Information

To deploy your HDP installation, you need to collect the following information:

- The fully qualified domain name (FQDN) for each host in your system, and which component(s) you wish to set up on which host. You can use `hostname -f` to check for the FQDN if you do not know it.
- The hostname (for an existing instance), database name, username, and password for the MySQL instance, if you install Hive/HCatalog.

Note

If you are using an existing instance, the dbuser you create for HDP's use must be granted ALL PRIVILEGES on that instance.

1.6. Prepare the Environment

To deploy your HDP instance, you need to prepare your deploy environment:

- Enable NTP on the Cluster
- Check DNS
- Disable SELinux
- Disable IPTables

1.6.1. Enable NTP on the Cluster

The clocks of all the nodes in your cluster must be able to synchronize with each other. If your system does not have access to the Internet, set up a master node as an NTP xserver. Use the following instructions to enable NTP for your cluster:

1. Configure NTP clients. Execute the following command on all the nodes in your cluster:

   ```
   yum install ntp
   ```

2. Enable the service. Execute the following command on all the nodes in your cluster:

   ```
   chkconfig ntpd on
   ```

3. Start the NTP. Execute the following command on all the nodes in your cluster:

   ```
   /etc/init.d/ntpd start
   ```

4. For using existing NTP server in your environment. Configure firewall on local NTP server to enable UDP input traffic on port 123. See the following sample rule:

   ```
   -A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -p udp --dport
    123 -j ACCEPT
   ```

   Restart iptables. Execute the following command on all the nodes in your cluster:

   ```
   service iptables restart
   ```

   Configure clients to use the local NTP server. Edit the `/etc/ntp.conf` and add the following line:

   ```
   server $LOCAL_SERVER_IP OR HOSTNAME
   ```

1.6.2. Check DNS

All hosts in your system must be configured for DNS and Reverse DNS.

Note

If you are unable to configure DNS and Reverse DNS, you must edit the hosts file on every host in your cluster to contain each of your hosts.

Use the following instructions to check DNS for all the host machines in your cluster:

1. Forward lookup checking.

   For example, for domain `localdomain` that contains host with name `host01` and IP address `192.168.0.10`, execute the following command:

   ```
   nslookup host01
   ```

   You should see a message similar to the following:

   ```
   Name: host01.localdomain
   Address: 192.168.0.10
   ```

2. Reverse lookup checking.

   For example, for domain `localdomain` that contains host with name `host01` and IP address `192.168.0.10`, execute the following command:

   ```
   nslookup 192.168.0.10
   ```

   You should see a message similar to the following:

   ```
   10.0.168.192.in-addr.arpa name = host01.localdomain.
   ```

If you do not receive valid responses (as shown above), you should set up DNS zone in your cluster or configure host files on each host of the cluster using one of the following options:

- **Option I:** Configure hosts file on each node of the cluster.

  For all nodes of cluster, add to the `/etc/hosts` file key-value pairs like the following:

  ```
  192.168.0.11 host01
  ```

- **Option II:** Configuring DNS using BIND nameserver.

  The following instructions, use the example values given below:

  ```
  Example values:
  domain name: “localdomain”
  nameserver: “host01”/192.168.0.11
  hosts: “host02”/192.168.0.12, “host02”/192.168.0.12
  ```

  1. Install BIND packages:

     ```
     yum install bind
     yum install bind-libs
     yum install bind-utils
     ```

2. Initiate service

```
chkconfig named on
```

3. Configure files. Add the following lines for the example values given above (ensure that you modify these for your environment) :

   - Edit the `/etc/resolv.conf` (for all nodes in cluster) and add the following lines:

```
domain localdomain
search localdomain
nameserver 192.168.0.11
```

   - Edit the `/etc/named.conf` (for all nodes in cluster) and add the following lines:

```
listen-on port 53 { any; };//by default it is opened only for localhost
 ...
zone "localdomain" {
 type master;
 notify no;
 allow-query { any; };
 file "named-forw.zone";
 };
 zone "0.168.192.in-addr.arpa" {
  type master;
  notify no;
  allow-query { any; };
  file "named-rev.zone";
};
```

   - Edit the `named-forw.zone` as shown in the following sample forward zone configuration file:

```
$TTL 3D
@ SOA   host01.localdomain.root.localdomain
(201306030;3600;3600;3600;3600)
NS host01            ; Nameserver Address
localhost IN A 127.0.0.1
host01  IN A 192.168.0.11
host02  IN A 192.168.0.12
host03  IN A 192.168.0.13
```

   - Edit the `named-rev.zone` as shown in the following sample reverse zone configuration file:

```
$TTL 3D
@ SOA host01.localdomain.root.localdomain. (201306031;28800;2H;4W;1D);
NS host01.localdomain.; Nameserver Address
11 IN PTR host01.localdomain.
12 IN PTR host02.localdomain.
13 IN PTR host03.localdomain.
```

4. Restart bind service.

```
/etc/init.d/named restart
```

5. Add rules to firewall.

```
iptables -A INPUT -p udp -m state --state NEW --dport 53 -j ACCEPT
iptables -A INPUT -p tcp -m state --state NEW --dport 53 -j ACCEPT
service iptables save
service iptables restart
```

Alternatively, you can also allow traffic over DNS port (53) using `system-config-firewall` utility.

1.6.3. Disable SELinux

Security-Enhanced (SE) Linux feature should be disabled during installation process.

1. Check state of SELinux. On all the host machines, execute the following command:

   ```
   getenforce
   ```

   If the result is `permissive` or `disabled`, no further actions are required, else proceed to step 2.

2. Disable SELinux either temporarily for each session or permanently.

   - **Option I:** Disable SELinux temporarily by executing the following command:

     ```
     setenforce 0
     ```

   - **Option II:** Disable SELinux permanently in the `/etc/sysconfig/selinux` file by changing the value of `SELINUX` field to `permissive` or `disabled`. Restart your system.

1.6.4. Disable IPTables

On all the RHEL/CentOS host machines, execute the following command to disable IPTables:

```
chkconfig iptables off
       /etc/init.d/iptables stop
```

1.7. [Optional] Install MySQL

If you are installing Hive and HCatalog services, you need a MySQL database instance to store metadata information. You can either use an existing MySQL instance or install a new instance of MySQL manually. To install a new instance:

1. Connect to the host machine you plan to use for Hive and HCatalog.

2. Install MySQL server. From a terminal window, type:

   For RHEL/CentOS: [6.x only supported for Beta]

   ```
   yum install mysql-server
   ```

3. Start the instance.

   ```
   /etc/init.d/mysqld start
   ```

4. Set the `root` user password.

```
mysqladmin -u root password ‘{password}’
```

5. Remove unnecessary information from log and STDOUT.

```
mysqladmin -u root 2>&1 >/dev/null
```

6. As `root`, use mysql (or other client tool) to create the “dbuser” and grant it adequate privileges. This user provides access to the Hive metastore.

```
CREATE USER '$dbusername'@'%' IDENTIFIED BY '$dbuserpassword';
GRANT ALL PRIVILEGES ON *.* TO '$dbusername'@'%';
flush privileges;
```

7. See if you can connect to the database as that user. You are prompted to enter the *`$dbuserpassword`* password above.

```
mysql -u $dbusername -p
```

8. Install the MySQL connector JAR file:

```
yum install mysql-connector-java*
```

1.8. Download Companion Files

We have provided a set of companion files, including script files (`scripts.zip`) and configuration files (`configuration_files.zip`), that you should download and use throughout this process. Download and extract the files:

```
wget http://public-repo-1.hortonworks.com/HDP/tools/2.0.5.0/
hdp_manual_install_rpm_helper_files-2.0.5.67.tar.gz
```

Hortonworks strongly recommends that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

The following provides a snapshot of a sample script file to create Hadoop directories. This sample script file sources the files included in Companion Files.

```
#!/bin/bashsource
./users.shsource
./directories.sh

echo "Create datanode local dir"
mkdir -p $DFS_DATA_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_DATA_DIR;
chmod -R 750 $DFS_DATA_DIR;

echo "Create yarn local dir"
mkdir -p $YARN_LOCAL_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_DIR;
chmod -R 755 $YARN_LOCAL_DIR;

echo "Create yarn local log dir"
mkdir -p $YARN_LOCAL_LOG_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_LOG_DIR;
chmod -R 755 $YARN_LOCAL_LOG_DIR;
```

1.9. Define Environment Parameters

You need to set up specific users and directories for your HDP installation using the following instructions:

1. Define directories.

   The following table describes the directories for install, configuration, data, process IDs and logs based on the Hadoop Services you plan to install. Use this table to define what you are going to use in setting up your environment.

Note

The `scripts.zip` file you downloaded in Download Companion Files includes a script, `directories.sh`, for setting directory environment parameters.

We strongly suggest you edit and source (alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

Table 1.1. Define Directories for Core Hadoop

Hadoop Service	Parameter	Definition

HDFS	`DFS_NAME_DIR`	Space separated list of directories where NameNode should store the file system image.
		
		For example,
		
		`/grid/hadoop/hdfs/nn`
		
		`/grid1/hadoop/hdfs/nn`

HDFS	`DFS_DATA_DIR`	Space separated list of directories where DataNodes should store the blocks.
		
		For example,
		
		`/grid/hadoop/hdfs/dn`
		
		`/grid1/hadoop/hdfs/dn`
		
		`/grid2/hadoop/hdfs/dn`

HDFS	`FS_CHECKPOINT_DIR`	Space separated list of directories where SecondaryNameNode should store the checkpoint image.
		
		For example,
		
		`/grid/hadoop/hdfs/snn`
		
		`/grid1/hadoop/hdfs/snn`
		
		`/grid2/hadoop/hdfs/snn`

HDFS	`HDFS_LOG_DIR`	Directory for storing the HDFS logs. This directory name is a combination of a directory and the *$HDFS_USER*. For example,

Hadoop Service	Parameter	Definition
		`/var/log/hadoop/hdfs` where `hdfs` is the *$HDFS_USER*.
HDFS	`HDFS_PID_DIR`	Directory for storing the HDFS process ID. This directory name is a combination of a directory and the *$HDFS_USER*. For example, `/var/run/hadoop/hdfs` where `hdfs` is the *$HDFS_USER*
HDFS	`HADOOP_CONF_DIR`	Directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`

YARN	`YARN_LOCAL_DIR`	Space separated list of directories where YARN should store temporary data.
		
		For example,
		
		`/grid/hadoop/yarn`
		
		`/grid1/hadoop/yarn`
		
		`/grid2/hadoop/yarn`.

YARN	`YARN_LOG_DIR`	Directory for storing the YARN logs. For example, `/var/log/hadoop/yarn`. This directory name is a combination of a directory and the *$YARN_USER*. In the example `yarn` is the *$YARN_USER*.
YARN	`YARN_PID_DIR`	Directory for storing the YARN process ID. For example, `/var/run/hadoop/yarn`. This directory name is a combination of a directory and the *$YARN_USER*. In the example, `yarn` is the *$YARN_USER*.
MapReduce	`MAPREDUCE_LOG_DIR`	Directory for storing the JobHistory Server logs. For example, `/var/log/hadoop/mapred`. This directory name is a combination of a directory and the *$MAPRED_USER*. In the example `mapred` is the *$MAPRED_USER*

Table 1.2. Define Directories for Ecosystem Components

Hadoop Service	Parameter	Definition
Pig	PIG_CONF_DIR	Directory to store the Pig configuration files. For example, /etc/pig/conf.
Pig	PIG_LOG_DIR	Directory to store the Pig logs. For example, /var/log/pig.
Pig	PIG_PID_DIR	Directory to store the Pig process ID. For example, /var/run/pig.
Oozie	OOZIE_CONF_DIR	Directory to store the Oozie configuration files. For example, /etc/oozie/conf.
Oozie	OOZIE_DATA	Directory to store the Oozie data. For example, /var/db/oozie.
Oozie	OOZIE_LOG_DIR	Directory to store the Oozie logs. For example, /var/log/oozie.
Oozie	OOZIE_PID_DIR	Directory to store the Oozie process ID. For example, /var/run/oozie.
Oozie	OOZIE_TMP_DIR	Directory to store the Oozie temporary files. For example, /var/tmp/oozie.
Hive	HIVE_CONF_DIR	Directory to store the Hive configuration files. For example, /etc/hive/conf.
Hive	HIVE_LOG_DIR	Directory to store the Hive logs. For example, /var/log/hive.
Hive	HIVE_PID_DIR	Directory to store the Hive process ID. For example, /var/run/hive.
WebHCat	WEBHCAT_CONF_DIR	Directory to store the WebHCat configuration files. For example, /etc/hcatalog/conf/webhcat.
WebHCat	WEBHCAT_LOG_DIR	Directory to store the WebHCat logs. For example, var/log/webhcat.
WebHCat	WEBHCAT_PID_DIR	Directory to store the WebHCat process ID. For example, /var/run/webhcat.
HBase	HBASE_CONF_DIR	Directory to store the HBase configuration files. For example, /etc/hbase/conf.
HBase	HBASE_LOG_DIR	Directory to store the HBase logs. For example, /var/log/hbase.
HBase	HBASE_PID_DIR	Directory to store the HBase process ID. For example, /var/run/hbase.
ZooKeeper	ZOOKEEPER_DATA_DIR	Directory where ZooKeeper will store data. For example, /grid/hadoop/zookeeper/data
ZooKeeper	ZOOKEEPER_CONF_DIR	Directory to store the ZooKeeper configuration files. For example, /etc/zookeeper/conf.
ZooKeeper	ZOOKEEPER_LOG_DIR	Directory to store the ZooKeeper logs. For example, /var/log/zookeeper.

Hadoop Service	Parameter	Definition
ZooKeeper	ZOOKEEPER_PID_DIR	Directory to store the ZooKeeper process ID. For example, /var/run/zookeeper.
Sqoop	SQOOP_CONF_DIR	Directory to store the Sqoop configuration files. For example, /usr/lib/sqoop/conf.

If you use the Companion files, the following provides a snapshot of how your directories.sh file should look after you edit the TODO variables:

```
#!/bin/sh

#
# Directories Script
#
# 1. To use this script, you must edit the TODO variables below for your
 environment.
#
# 2. Warning: Leave the other parameters as the default values. Changing
 these default values will require you to
# change values in other configuration files.
#

#
# Hadoop Service - HDFS
#

# Space separated list of directories where NameNode will store file system
 image. For example, /grid/hadoop/hdfs/nn /grid1/hadoop/hdfs/nn
DFS_NAME_DIR="/grid/0/hadoop/hdfs/nn";

# Space separated list of directories where DataNodes will store the blocks.
 For example, /grid/hadoop/hdfs/dn /grid1/hadoop/hdfs/dn /grid2/hadoop/hdfs/
dn
DFS_DATA_DIR="/grid/0/hadoop/hdfs/dn";

# Space separated list of directories where SecondaryNameNode will store
 checkpoint image. For example, /grid/hadoop/hdfs/snn /grid1/hadoop/hdfs/
snn /grid2/hadoop/hdfs/snn
FS_CHECKPOINT_DIR="/grid/0/hadoop/hdfs/snn";



# Directory to store the HDFS logs.
HDFS_LOG_DIR="/var/log/hadoop/hdfs";

# Directory to store the HDFS process ID.
HDFS_PID_DIR="/var/run/hadoop/hdfs";

# Directory to store the Hadoop configuration files.
HADOOP_CONF_DIR="/etc/hadoop/conf";

#
# Hadoop Service - YARN
#
```

```
# Space separated list of directories where YARN will store temporary data.
 For example, /grid/hadoop/yarn/local /grid1/hadoop/yarn/local /grid2/
hadoop/yarn/local
YARN_LOCAL_DIR="/grid/0/hadoop/yarn/local";

# Directory to store the YARN logs.
YARN_LOG_DIR="/var/log/hadoop/yarn";

# Space separated list of directories where YARN will store container log
 data. For example, /grid/hadoop/yarn/logs /grid1/hadoop/yarn/logs /grid2/
hadoop/yarn/logs
YARN_LOCAL_LOG_DIR="/grid/0/hadoop/yarn/logs";

# Directory to store the YARN process ID.
YARN_PID_DIR="/var/run/hadoop/yarn";

#
# Hadoop Service - MAPREDUCE
#

# Directory to store the MapReduce daemon logs.
MAPRED_LOG_DIR="/var/log/hadoop/mapreduce";

# Directory to store the mapreduce jobhistory process ID.
MAPRED_PID_DIR="/var/run/hadoop/mapreduce";

#
# Hadoop Service - Hive
#

# Directory to store the Hive configuration files.
HIVE_CONF_DIR="/etc/hive/conf";

# Directory to store the Hive logs.
HIVE_LOG_DIR="/var/log/hive";

# Directory to store the Hive process ID.
HIVE_PID_DIR="/var/run/hive";

#
# Hadoop Service - WebHCat (Templeton)
#

# Directory to store the WebHCat (Templeton) configuration files.
WEBHCAT_CONF_DIR="/etc/hcatalog/conf/webhcat";

# Directory to store the WebHCat (Templeton) logs.
WEBHCAT_LOG_DIR="var/log/webhcat";

# Directory to store the WebHCat (Templeton) process ID.
WEBHCAT_PID_DIR="/var/run/webhcat";

#
# Hadoop Service - HBase
#

# Directory to store the HBase configuration files.
HBASE_CONF_DIR="/etc/hbase/conf";

# Directory to store the HBase logs.
```

```
HBASE_LOG_DIR="/var/log/hbase";

# Directory to store the HBase logs.
HBASE_PID_DIR="/var/run/hbase";

#
# Hadoop Service - ZooKeeper
#

# Directory where ZooKeeper will store data. For example, /grid1/hadoop/
zookeeper/data
ZOOKEEPER_DATA_DIR="../hadoop/zookeeper/data";

# Directory to store the ZooKeeper configuration files.
ZOOKEEPER_CONF_DIR="/etc/zookeeper/conf";

# Directory to store the ZooKeeper logs.
ZOOKEEPER_LOG_DIR="/var/log/zookeeper";

# Directory to store the ZooKeeper process ID.
ZOOKEEPER_PID_DIR="/var/run/zookeeper";

#
# Hadoop Service - Pig
#

# Directory to store the Pig configuration files.
PIG_CONF_DIR="/etc/pig/conf";

# Directory to store the Pig logs.
PIG_LOG_DIR="/var/log/pig";

# Directory to store the Pig process ID.
PIG_PID_DIR="/var/run/pig";


#
# Hadoop Service - Oozie
#

# Directory to store the Oozie configuration files.
OOZIE_CONF_DIR="/etc/oozie/conf"

# Directory to store the Oozie data.
OOZIE_DATA="/var/db/oozie"

# Directory to store the Oozie logs.
OOZIE_LOG_DIR="/var/log/oozie"

# Directory to store the Oozie process ID.
OOZIE_PID_DIR="/var/run/oozie"

# Directory to store the Oozie temporary files.
OOZIE_TMP_DIR="/var/tmp/oozie"

#
# Hadoop Service - Sqoop
#
SQOOP_CONF_DIR="/etc/sqoop/conf"
```

```
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
```

2. Define users and groups:

   The following table describes system user account and groups. Use this table to define what you are going to use in setting up your environment. These users and groups should reflect the accounts you created in Create System Users and Groups.

Note

The `scripts.zip` file you downloaded in Download Companion Files includes a script, `usersAndGroups.sh`, for setting user and group environment parameters.

We strongly suggest you edit and source (alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

Table 1.3. Define Users and Groups for Systems

Parameter	Definition
HDFS_USER	User owning the HDFS services. For example, `hdfs`.
YARN_USER	User owning the YARN services. For example, `yarn`.
ZOOKEEPER_USER	User owning the ZooKeeper services. For example, `zookeeper`.
HIVE_USER	User owning the Hive services. For example, `hive`.
WEBHCAT_USER	User owning the WebHCat services. For example, `hcat`.
HBASE_USER	User owning the HBase services. For example, `hbase`.
PIG_USER	User owning the Pig services. For example, `pig`.
HADOOP_GROUP	A common group shared by services. For example, `hadoop`.

1.10. [Optional] Create System Users and Groups

In general Hadoop services should be owned by specific users and not by root or application users.The table below shows the typical users for Hadoop services. If you choose to install the HDP components using the RPMs, these users will automatically be set up.

If you do not install with the RPMs, or want different users, then you must identify the users that you want for your Hadoop services and the common Hadoop group and create these accounts on your system.

Table 1.4. Typical System Users and Groups

Hadoop Service	User	Group
HDFS	hdfs	hadoop
YARN	yarn	hadoop
MapReduce	mapred	hadoop
Hive	hive	hadoop
Pig	pig	hadoop
HCatalog/WebHCatalog	hcat	hadoop

Hadoop Service	User	Group
HBase	hbase	hadoop
ZooKeeper	zookeeper	hadoop
Oozie	oozie	hadoop

2. Installing HDFS and YARN

This section describes how to install the Hadoop Core components, HDFS, YARN, and MapReduce.

Complete the following instructions to install Hadoop Core components:

1. Set Default File and Directory Permissions
2. Install the Hadoop RPMs
3. Install Compression Libraries
4. Create Directories

2.1. Set Default File and Directory Permissions

Set the default file and directory permissions to 0022 (022). This is typically the default for most Linux distributions.

Use the `umask` command to confirm and set as necessary.

Ensure that the `umask` is set for all terminal sessions that you use during installation.

2.2. Install the Hadoop RPMs

Execute the following command on all cluster nodes.

From a terminal window, type:

```
yum install hadoop hadoop-hdfs hadoop-libhdfs hadoop-yarn hadoop-mapreduce
 hadoop-client openssl
```

2.3. Install Compression Libraries

Make the following compression libraries available on all the cluster nodes.

2.3.1. Install Snappy

Complete the following instructions on all the nodes in your cluster:

1. Install Snappy.

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

2. Make the Snappy libraries available to Hadoop:

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

2.3.2. Install LZO

Execute the following command on all the nodes in your cluster. From a terminal window, type:

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

2.4. Create Directories

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them.

Use the following instructions to create appropriate directories:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Create the NameNode directories
3. Create the Secondary NameNode directories
4. Create the DataNode and YARN NodeManager local directories
5. Create the log and PID directories

2.4.1. Create the NameNode Directories

On the node that hosts the NameNode service, execute the following commands:

```
mkdir -p $DFS_NAME_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_NAME_DIR;
chmod -R 755 $DFS_NAME_DIR;
```

where:

- *$DFS_NAME_DIR* is the space separated list of directories where NameNode stores the file system image. For example, `/grid/hadoop/hdfs/nn /grid1/hadoop/hdfs/nn`.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

2.4.2. Create the SecondaryNameNode Directories

On all the nodes that can potentially run the SecondaryNameNode service, execute the following commands:

```
mkdir -p $FS_CHECKPOINT_DIR;
```

```
chown -R $HDFS_USER:$HADOOP_GROUP $FS_CHECKPOINT_DIR;
chmod -R 755 $FS_CHECKPOINT_DIR;
```

where:

- *$FS_CHECKPOINT_DIR* is the space separated list of directories where SecondaryNameNode should store the checkpoint image. For example, `/grid/hadoop/hdfs/snn /grid1/hadoop/hdfs/snn /grid2/hadoop/hdfs/snn`.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

2.4.3. Create DataNode and YARN NodeManager Local Directories

On all DataNodes, execute the following commands:

```
mkdir -p $DFS_DATA_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_DATA_DIR;
chmod -R 750 $DFS_DATA_DIR;
```

where:

- *$DFS_DATA_DIR* is the space separated list of directories where DataNodes should store the blocks. For example, `/grid/hadoop/hdfs/dn /grid1/hadoop/hdfs/dn /grid2/hadoop/hdfs/dn`.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

On the ResourceManager and all Datanodes, execute the following commands:

```
mkdir -p $YARN_LOCAL_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_DIR;
chmod -R 755 $YARN_LOCAL_DIR;
```

where:

- *$YARN_LOCAL_DIR* is the space separated list of directories where YARN should store temporary data. For example, `/grid/hadoop/yarn/local /grid1/hadoop/yarn/local /grid2/hadoop/yarn/local`.
- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

On the ResourceManager and all Datanodes, execute the following commands:

```
mkdir -p $YARN_LOCAL_LOG_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_LOG_DIR;
chmod -R 755 $YARN_LOCAL_LOG_DIR;
```

where:

- *$YARN_LOCAL_LOG_DIR* is the space separated list of directories where YARN should store temporary data. For example, `/grid/hadoop/yarn/logs /grid1/hadoop/yarn/logs /grid2/hadoop/yarn/local`.
- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

2.4.4. Create the Log and PID Directories

On all nodes, execute the following commands:

```
mkdir -p $HDFS_LOG_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $HDFS_LOG_DIR;
chmod -R 755 $HDFS_LOG_DIR;
```

where:

- *$HDFS_LOG_DIR* is the directory for storing the HDFS logs.

  This directory name is a combination of a directory and the *$HDFS_USER*.

  For example, `/var/log/hadoop/hdfs` where `hdfs` is the *$HDFS_USER*.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

```
mkdir -p $YARN_LOG_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOG_DIR;
chmod -R 755 $YARN_LOG_DIR;
```

where:

- *$YARN_LOG_DIR* is the directory for storing the YARN logs.

  This directory name is a combination of a directory and the *$YARN_USER*.

  For example, `/var/log/hadoop/yarn` where `yarn` is the *$YARN_USER*.
- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

```
mkdir -p $HDFS_PID_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $HDFS_PID_DIR;
chmod -R 755 $HDFS_PID_DIR
```

where:

- *$HDFS_PID_DIR* is the directory for storing the HDFS process ID.

  This directory name is a combination of a directory and the *$HDFS_USER*.

  For example, `/var/run/hadoop/hdfs` where `hdfs` is the *$HDFS_USER*.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.

- *$HADOOP_GROUP* is a common group shared by services. For example, hadoop.

```
mkdir -p $YARN_PID_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_PID_DIR;
chmod -R 755 $YARN_PID_DIR;
```

where:

- *$YARN_PID_DIR* is the directory for storing the YARN process ID.

  This directory name is a combination of a directory and the *$YARN_USER*.

  For example, /var/run/hadoop/yarn where yarn is the *$YARN_USER*.

- *$YARN_USER* is the user owning the YARN services. For example, yarn.

- *$HADOOP_GROUP* is a common group shared by services. For example, hadoop.

```
mkdir -p $MAPRED_LOG_DIR;
chown -R $MAPRED_USER:$HADOOP_GROUP $MAPRED_LOG_DIR;
chmod -R 755 $MAPRED_LOG_DIR;
```

where:

- *$MAPRED_LOG_DIR* is the directory for storing the JobHistory Server logs.

  This directory name is a combination of a directory and the *$MAPREDs_USER*.

  For example, /var/log/hadoop/mapred where mapred is the *$MAPRED_USER*.

- *$MAPRED_USER* is the user owning the MAPRED services. For example, mapred.

- *$HADOOP_GROUP* is a common group shared by services. For example, hadoop.

```
mkdir -p $MAPRED_PID_DIR;
chown -R $MAPRED_USER:$HADOOP_GROUP $MAPRED_PID_DIR;
chmod -R 755 $MAPRED_PID_DIR;
```

where:

- *$MAPRED_PID_DIR* is the directory for storing the JobHistory Server pid.

  This directory name is a combination of a directory and the *$MAPREDs_USER*.

  For example, /var/run/hadoop/mapred where mapred is the *$MAPRED_USER*.

- *$MAPRED_USER* is the user owning the MAPRED services. For example, mapred.

- *$HADOOP_GROUP* is a common group shared by services. For example, hadoop.

3. Setting Up the Hadoop Configuration

This section describes how to set up and edit the deployment configuration files for HDFS and MapReduce.

Use the following instructions to set up Hadoop configuration files:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. From the downloaded `scripts.zip` file, extract the files from the `configuration_files/core_hadoop` directory to a temporary directory.

3. Modify the configuration files.

   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. See Define Environment Parameters for more information.

   a. Edit the `core-site.xml` and modify the following properties:

```
<property>
 <name>fs.defaultFS</name>
 <value>hdfs://$namenode.full.hostname:8020</value>
 <description>Enter your NameNode hostname</description>
</property>
```

   b. Edit the `hdfs-site.xml` and modify the following properties:

```
<property>
 <name>dfs.namenode.name.dir</name>
 <value>/grid/hadoop/hdfs/nn,/grid1/hadoop/hdfs/nn</value>
 <description>Comma separated list of paths. Use the list of directories
 from $DFS_NAME_DIR.
                For example, /grid/hadoop/hdfs/nn,/grid1/hadoop/hdfs/nn.
</description>
</property>
```

```
<property>
 <name>dfs.datanode.data.dir</name>
 <value>/grid/hadoop/hdfs/dn,/grid1/hadoop/hdfs/dn</value>
 <description>Comma separated list of paths. Use the list of directories
 from $DFS_DATA_DIR.
                For example, /grid/hadoop/hdfs/dn,/grid1/hadoop/hdfs/dn.
</description>
                </property>
```

```
<property>
 <name>dfs.namenode.http-address</name>
 <value>$namenode.full.hostname:50070</value>
 <description>Enter your NameNode hostname for http access.</description>
</property>
```

```
<property>
 <name>dfs.namenode.secondary.http-address</name>
 <value>$secondary.namenode.full.hostname:50090</value>
 <description>Enter your Secondary NameNode hostname.</description>
 </property>
```

```
<property>
   <name>dfs.namenode.checkpoint.dir</name>
    <value>/grid/hadoop/hdfs/snn,/grid1/hadoop/hdfs/snn,/grid2/hadoop/
hdfs/snn</value>
     <description>A comma separated list of paths. Use the list of
 directories from $FS_CHECKPOINT_DIR.
     For example, /grid/hadoop/hdfs/snn,sbr/grid1/hadoop/hdfs/snn,sbr/
grid2/hadoop/hdfs/snn </description>
     </property>
```

Note

The value of NameNode new generation size should be 1/8 of maximum heap size (`-Xmx`). Ensure that you check the default setting for your environment.

To change the default value:

i. Edit the `/etc/hadoop/conf/hadoop-env.sh` file.

ii. Change the value of the `-XX:MaxnewSize` parameter to 1/8th the value of the maximum heap size (`-Xmx`) parameter.

c. Edit the `yarn-site.xml` and modify the following properties:

```
<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value>$resourcemanager.full.hostname:8025</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>$resourcemanager.full.hostname:8030</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.address</name>
 <value>$resourcemanager.full.hostname:8050</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.admin.address</name>
 <value>$resourcemanager.full.hostname:8041</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.nodemanager.local-dirs</name>
 <value>/grid/hadoop/hdfs/yarn/local,/grid1/hadoop/hdfs/yarn/local</
value>
 <description>Comma separated list of paths. Use the list of directories
 from $YARN_LOCAL_DIR.
               For example, /grid/hadoop/hdfs/yarn/local,/grid1/hadoop/
hdfs/yarn/local.</description>
</property>
```

```
<property>
 <name>yarn.nodemanager.log-dirs</name>
 <value>/gird/hadoop/hdfs/yarn/logs</value>
 <description>Use the list of directories from $YARN_LOCAL_LOG_DIR.
               For example, /grid/hadoop/yarn/logs /grid1/hadoop/yarn/
logs /grid2/hadoop/yarn/logs</description>
</property>
```

```
<property>
 <name>yarn.log.server.url</name>
 <value>http://$jobhistoryserver.full.hostname:19888/jobhistory/logs/</
value>
 <description>URL for job history server</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.webapp.address</name>
 <value>$resourcemanager.full.hostname:8088</value>
 <description>URL for job history server</description>
</property>
```

d. Edit the `mapred-site.xml` and modify the following properties:

```
<property>
 <name>mapreduce.jobhistory.address</name>
 <value>$jobhistoryserver.full.hostname:10020</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>
```

```
<property>
 <name>mapreduce.jobhistory.webapp.address</name>
 <value>$jobhistoryserver.full.hostname:19888</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>
```

4. Optional: Configure MapReduce to use Snappy Compression

   In order to enable Snappy compression for MapReduce jobs, edit core-site.xml and mapred-site.xml.

   a. Add the following properties to mapred-site.xml:

```
<property>
<name>mapreduce.admin.map.child.java.opts</name>
```

```
<value>-server -XX:NewRatio=8 -Djava.library.path=/usr/lib/hadoop/lib/
native/ -Djava.net.preferIPv4Stack=true</value>
<final>true</final>
</property>
<property>
<name>mapreduce.admin.reduce.child.java.opts</name>
<value>-server -XX:NewRatio=8 -Djava.library.path=/usr/lib/hadoop/lib/
native/ -Djava.net.preferIPv4Stack=true</value>
<final>true</final>
</property>
```

b. Add the SnappyCodec to the codecs list in core-site.xml:

```
<property>
<name>io.compression.codecs</name>
<value>org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.
compress.DefaultCodec,org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

5. Copy the configuration files.

a. On all hosts in your cluster, create the Hadoop configuration directory:

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

where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files.

For example, `/etc/hadoop/conf`.

b. Copy all the configuration files to *$HADOOP_CONF_DIR*.

c. Set appropriate permissions:

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

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

4. Validating the Core Hadoop Installation

Use the following instructions to start core Hadoop and perform the smoke tests:

1. Format and Start HDFS
2. Smoke Test HDFS
3. Start YARN
4. Start MapReduce JobHistory Server
5. Smoke Test MapReduce

4.1. Format and Start HDFS

1. Execute these commands on the NameNode host machine:

```
su $HDFS_USER
/usr/lib/hadoop/bin/hadoop namenode -format
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
 namenode
```

2. Execute these commands on the SecondaryNameNode:

```
su $HDFS_USER
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
 secondarynamenode
```

3. Execute these commands on all DataNodes:

```
su $HDFS_USER
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
 datanode
```

where:

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

4.2. Smoke Test HDFS

1. See if you can reach the NameNode server with your browser:

```
http://$namenode.full.hostname:50070
```

2. Create hdfs user directory in HDFS:

```
su $HDFS_USER
```

```
hadoop fs -mkdir -p /user/hdfs
```

3. Try copying a file into HDFS and listing that file:

```
su $HDFS_USER
hadoop fs -copyFromLocal /etc/passwd passwd
hadoop fs -ls
```

4. Test browsing HDFS:

```
http://$datanode.full.hostname:50075/browseDirectory.jsp?namenodeInfoPort=
50070&dir=/&nnaddr=$datanode.full.hostname:8020
```

4.3. Start YARN

1. Execute these commands from the ResourceManager server:

```
<login as $YARN_USER>
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start
 resourcemanager
```

2. Execute these commands from all NodeManager nodes:

```
<login as $YARN_USER>
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start
 nodemanager
```

where:

- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.
- *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

4.4. Start MapReduce JobHistory Server

1. Execute these commands from the JobHistory server to set up directories on HDFS :

```
su $HDFS_USER
hadoop fs -mkdir -p /mr-history/tmp
hadoop fs -chmod -R 1777 /mr-history/tmp
hadoop fs -mkdir -p /mr-history/done
hadoop fs -chmod -R 1777 /mr-history/done
hadoop fs -chown -R $MAPRED_USER:$HDFS_USER /mr-history

hadoop fs -mkdir -p /app-logs
hadoop fs -chmod -R 1777 /app-logs
hadoop fs -chown yarn /app-logs
```

2. Execute these commands from the JobHistory server:

```
<login as $MAPRED_USER>
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec/
/usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh --
config $HADOOP_CONF_DIR start historyserver
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$MAPRED_USER* is the user owning the MapRed services. For example, `mapred`.
- *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

4.5. Smoke Test MapReduce

1. Try browsing to the ResourceManager:

```
http://$resourcemanager.full.hostname:8088/
```

2. Smoke test using Terasort and sort 10GB of data.

```
su $HDFS_USER
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-2*.jar teragen 100 /test/10gsort/input
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-2*.jar terasort /test/10gsort/input /test/10gsort/output
```

5. Installing Apache Pig

This section describes installing and testing Apache Pig, a platform for creating higher level data flow programs that can be compiled into sequences of MapReduce programs, using Pig Latin, the platform's native language.

Complete the following instructions to install Pig:

1. Install the Pig RPMs
2. Set Up Configuration Files
3. Validate the Installation

5.1. Install the Pig RPMs

On all the hosts where Pig programs will be executed, install the RPMs. From a terminal window, type:

```
yum install pig
```

The RPM will install Pig libraries to `/usr/lib/pig`. Pig configuration files are placed in `/usr/lib/pig/conf`.

5.2. Set Up Configuration Files

Use the following instructions to set up configuration files for Pig:

1. Extract the Pig configuration files.

   From the downloaded `scripts.zip` file, extract the files from the `configuration_files/pig` directory to a temporary directory.

2. Copy the configuration files.

   a. On all hosts where Pig will be executed, create the Pig configuration directory:

   ```
   rm -r $PIG_CONF_DIR
   mkdir -p $PIG_CONF_DIR
   ```

   b. Copy all the configuration files to *$PIG_CONF_DIR*.

   c. Set appropriate permissions:

   ```
   chown -R $PIG_USER:$HADOOP_GROUP $PIG_CONF_DIR/../
   chmod -R 755 $PIG_CONF_DIR/../
   ```

   where:

   - *$PIG_CONF_DIR* is the directory to store Pig configuration files. For example, `/etc/pig/conf`.
   - *$PIG_USER* is the user owning the Pig services. For example, `pig`.

- *$HADOOP_GROUP* is a common group shared by services. For example, hadoop.

5.3. Validate the Installation

Use the following steps to validate your installation:

1. On the host machine where Pig is installed execute the following commands:

```
login as $HDFS_USER
/usr/lib/hadoop/bin/hadoop dfs -copyFromLocal /etc/passwd passwd
```

2. Create the pig script file /tmp/id.pig with the following contents:

```
A = load 'passwd' using PigStorage(':');
B = foreach A generate \$0 as id; store B into '/tmp/id.out';
```

3. Execute the Pig script:

```
export JAVA_HOME=/usr/java/default
 pig -l /tmp/pig.log /tmp/id.pig
```

6. Installing Apache Hive and Apache HCatalog

This section describes installing and testing Apache Hive, a tool for creating higher level SQL queries using HiveQL, the tool's native language, that can be compiled into sequences of MapReduce programs.

It also describes installing and testing Apache HCatalog, a metadata abstraction layer that insulates users and scripts from how and where data is physically stored.

Complete the following instructions to install Hive and HCatalog:

1. Install the Hive and HCatalog RPMs
2. Set Directories and Permissions
3. Set Up the Hive/HCatalog Configuration Files
4. Create Directories on HDFS
5. Validate the Installation

6.1. Install the Hive and HCatalog RPMs

1. On all client/gateway nodes (on which Hive programs will be executed), Hive Metastore Server, and HiveServer2 machine, install the Hive RPMs.

   ```
   yum install hive hcatalog
   ```

2. Optional - Download and add the database connector JAR.

   By default, Hive uses embedded Derby database for its metastore. However, you can optionally choose to enable remote database (MySQL) for Hive metastore.

   a. Execute the following command on the Hive metastore machine.

      ```
      yum install mysql-connector-java*
      ```

   b. After the yum install, the mysql jar is placed in '/usr/share/java/'. Copy the downloaded JAR file to the `/usr/lib/hive/lib/` directory on your Hive host machine.

   c. Ensure that the JAR file has appropriate permissions.

6.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Pig configuration files :

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute these commands on the Hive server machine:

```
mkdir -p $HIVE_LOG_DIR;
```

```
chown -R $HIVE_USER:$HADOOP_GROUP $HIVE_LOG_DIR;
chmod -R 755 $HIVE_LOG_DIR;
```

   where:

   - *$HIVE_LOG_DIR* is the directory for storing theHive Server logs.

     This directory name is a combination of a directory and the *$HIVE_USER*.

   - *$HIVE_USER* is the user owning the Hive services. For example, `hive`.

   - *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

6.3. Set Up the Hive/HCatalog Configuration Files

Use the following instructions to set up the Hive/HCatalog configuration files:

1. Extract the Hive/HCatalog configuration files.

   From the downloaded `scripts.zip` file, extract the files in `configuration_files/hive` directory to a temporary directory.

2. Modify the configuration files.

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

   a. Edit `hive-site.xml` and modify the following properties:

```
<property>
 <name>javax.jdo.option.ConnectionURL</name>
 <value>jdbc:mysql://$mysql.full.hostname:3306/$database.name?
createDatabaseIfNotExist=true</value>
 <description>Enter your JDBC connection string. </description>
</property>
```

```
<property>
 <name>javax.jdo.option.ConnectionUserName</name>
 <value>$dbusername</value>
 <description>Enter your MySQL credentials. </description>
</property>
```

```
<property>
 <name>javax.jdo.option.ConnectionPassword</name>
 <value>$dbuserpassword</value>
 <description>Enter your MySQL credentials. </description>
</property>
```

Enter your MySQL credentials from Install MySQL (Optional).

```
<property>
 <name>hive.metastore.uris</name>
 <value>thrift://$metastore.server.full.hostname:9083</value>
 <description>URI for client to contact metastore server. To enable
 HiveServer2, leave the property value empty. </description>
</property>
```

3. Copy the configuration files.

   a. On all Hive hosts create the Hive configuration directory.

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

   b. Copy all the configuration files to *$HIVE_CONF_DIR* directory.

   c. Set appropriate permissions:

   ```
   chown -R $HIVE_USER:$HADOOP_GROUP $HIVE_CONF_DIR/../ ;
   chmod -R 755 $HIVE_CONF_DIR/../ ;
   ```

   where:

   - *$HIVE_CONF_DIR* is the directory to store the Hive configuration files. For example, `/etc/hive/conf`.
   - *$HIVE_USER* is the user owning the Hive services. For example, `hive`.
   - *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

6.4. Create Directories on HDFS

1. Create Hive user home directory on HDFS.

   ```
   Login as $HDFS_USER
   hadoop fs -mkdir -p /user/$HIVE_USER
   hadoop fs -chown $HIVE_USER:$HDFS_USER /user/$HIVE_USER
   ```

2. Create warehouse directory on HDFS.

   ```
   Login as $HDFS_USER
   hadoop fs -mkdir -p /apps/hive/warehouse
   hadoop fs -chown -R $HIVE_USER:$HDFS_USER /apps/hive
   hadoop fs -chmod -R 775 /apps/hive
   ```

   where:

   - *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
   - *$HIVE_USER* is the user owning the Hive services. For example, `hive`.

3. Create hive scratch directory on HDFS.

   ```
   Login as $HDFS_USER
   hadoop fs -mkdir -p /tmp/scratch
   hadoop fs -chown -R $HIVE_USER:$HDFS_USER /tmp/scratch
   ```

```
hadoop fs -chmod -R 777 /tmp/scratch
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HIVE_USER* is the user owning the Hive services. For example, `hive`.

6.5. Validate the Installation

Use the following steps to validate your installation:

1. Start Hive Metastore service.

```
 Login as $HIVE_USER
nohup hive --service metastore>$HIVE_LOG_DIR/hive.out 2>$HIVE_LOG_DIR/hive.
log &
```

2. Smoke Test Hive.

   a. Open Hive command line shell.

```
hive
```

   b. Run sample commands.

```
show databases;
create table test(col1 int, col2 string);
show tables;
```

3. Start HiveServer2.

```
 /usr/lib/hive/bin/hiveserver2 >$HIVE_LOG_DIR/hiveserver2.out
 2> $HIVE_LOG_DIR/hiveserver2.log &
```

4. Smoke Test HiveServer2.

   a. Open Beeline command line shell to interact with HiveServer2.

```
/usr/lib/hive/bin/beeline
```

   b. Establish connection to server.

```
!connect jdbc:hive2://$hive.server.full.hostname:10000 $HIVE_USER
 password org.apache.hive.jdbc.HiveDriver
```

   c. Run sample commands.

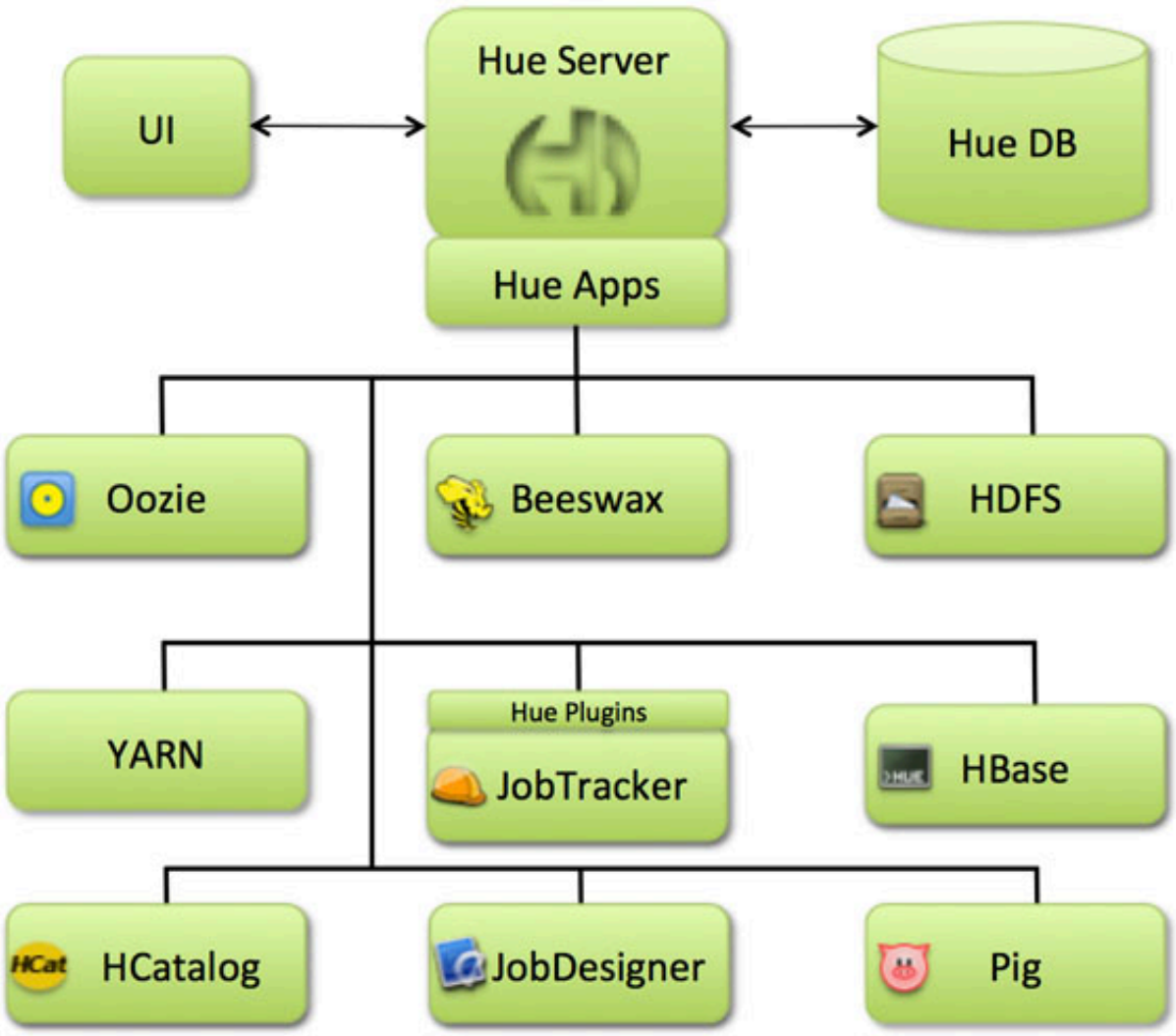
```
show databases;
create table test2(a int, b string);
show tables;
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HIVE_LOG_DIR* is the directory for storing theHive Server logs. This directory name is a combination of a directory and the *$HIVE_USER*.

7. Installing Hue

Hue provides a Web application interface for Apache Hadoop. It supports a file browser, JobTracker interface, Hive, Pig, Oozie, HBase, and more.



Complete the following instructions to install Apache Hue:

1. Prerequisites
2. Configure HDP
3. Install Hue
4. Configure Hue
5. Start Hue

7.1. Prerequisites

Complete the following prerequisites before deploying Hue:

1. For RHEL/CentOs 5.x verify that you are deploying the following dependency on all the host machines in your cluster:

   ```
   yum install python26
   ```

2. Stop all the services in your cluster. For more information see the instructions provided here.

3. Install and run Hadoop from HDP-2.0.5.0.

   The following table outlines the dependencies on the HDP components:

   Table 7.1. Dependencies on the HDP components

Component	Required	Applications	Notes
HDFS	Yes	Core, Filebrowser	HDFS access through WebHDFS or HttpFS
Yarn	Yes	JobDesigner, JobBrowser, Beeswax	Transitive dependency via Hive or Oozie
Oozie	No	JobDesigner, Oozie	Oozie access through REST API
Hive	No	Beeswax, HCatalog	Beeswax uses the Hive client libraries
WebHCat	No	HCatalog, Pig	HCatalog and Pig use WebHcat REST API
HBase	No	Shell	Optionally provides access to the HBase shell

4. Choose a Hue Server host machine in your cluster where you want to deploy Hue.

   Typically, you can choose to deploy Hue on any node within your cluster. However, if your corporate firewall policies allow, you can also use a remote host machine as your Hue server. For pilot or small cluster sizes, you can use the master install machine for HDP as your Hue server.

5. Configure the firewall.

   a. Verify that the host machines within your cluster can connect to each other over TCP.

   b. The machines outside your cluster must be able to open TCP port `8000` on the Hue Server (or the configured Hue web HTTP port) to interact with the system.

7.2. Configure HDP

1. Modify the `hdfs-site.xml` file.

   On the NameNode, Secondary NameNode, and all the DataNodes, add the following properties to the *`$HADOOP_CONF_DIR`*`/hdfs-site.xml` file.

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

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

2. Modify the `core-site.xml` file.

   On the NameNode, Secondary NameNode, and all the DataNodes, add the following properties to the *$HADOOP_CONF_DIR*`/core-site.xml` file.

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

```
<property>
  <name>hadoop.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>hadoop.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

3. Modify the webhcat-site.xml file. vi $WEBHCAT_CONF_DIR/webhcat-site.xml

```
<property>
  <name>webhcat.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>webhcat.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

4. Modify the oozie-site.xml file vi $OOZIE_CONF_DIR/oozie-site.xml

```
<property>
  <name>oozie.service.ProxyUserService.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>oozie.service.ProxyUserService.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

5. Stop the NameNode by running the following command:

```
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR stop
 namenode
```

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

7.3. Install Hue

Execute the following command on all Hue server host machines:

- For RHEL/CentOS: [6.x only supported for Beta]

```
yum install hue
```

- For SLES: [Not supported for Beta]

```
zypper install hue
```

7.4. Configure Hue

Use the following commands to explore the configuration options for Hue.

- To list all available configuration options:

```
/usr/lib/hue/build/env/bin/hue config_help | less
```

- To view current configuration options:

```
http://<hue>/dump_config
```

- To use multiple files to store your configuration:

  Hue loads and merges all of the files with extension `.ini` located in the `/etc/hue` directory.

Use the following instructions to configure Hadoop for Hue:

1. Configure Web Server
2. Configure Hadoop
3. Configure Beeswax
4. Configure JobDesigner and Oozie
5. Configure UserAdmin
6. Configure WebHCat

7.4.1. Configure Web Server

Use the following instructions to configure Web server:

These configuration variables are under the `[desktop]` section in the `hue.ini` configuration file.

1. Specify the Hue HTTP Address.

   Use the following options to change the IP address and port of the existing Web Server for Hue (by default, Spawning or CherryPy).

```
# Webserver listens on this address and port
http_host=0.0.0.0
http_port=8888
```

The default setting is port 8888 on all configured IP addresses.

2. Specify the Secret Key.

   To ensure that your session cookies are secure, enter a series of random characters (30 to 60 characters is recommended) as shown below:

   ```
   secret_key=jFE93j;2[290-eiw.KEiwN2s3['d;/.q[eIW^y#e=+Iei*@Mn<qW5o
   ```

3. Configure authentication.

   By default, the first user who logs in to Hue can choose any username and password and gets the administrator privileges. This user can create other user and administrator accounts. User information is stored in the Django database in the Django backend.

4. Configure Hue for SSL.

   Install pyOpenSSL in order to configure Hue to serve over HTTPS. To install pyOpenSSL, from the root of your Hue installation path, complete the following instructions:

   a. Execute the following command on the Hue Server:

      ```
      ./build/env/bin/easy_install pyOpenSSL
      ```

   b. Configure Hue to use your private key. Add the following to `hue.ini` file:

      ```
      ssl_certificate=$PATH_To_CERTIFICATE
      ssl_private_key=$PATH_To_KEY
      ```

      Ideally, you should have an appropriate key signed by a Certificate Authority. For test purposes, you can create a self-signed key using the `openssl` command on your system:

      ```
       ### Create a key
      openssl genrsa 1024 > host.key
      ```

      ```
      ### Create a self-signed certificate
      openssl req -new -x509 -nodes -sha1 -key host.key > host.cert
      ```

      Note

      To upload files using the Hue File Browser over HTTPS, you must have a proper SSL Certificate.

7.4.2. Configure Hadoop

Use the following instructions to configure Hadoop:

These configuration variables are under the `[hadoop]` section in the `hue.ini` configuration file.

1. Configure HDFS Cluster.

Hue supports only one HDFS cluster currently.

Ensure that you define the HDFS cluster under the `[[[default]]]` sub-section. Use the following variables to configure the HDFS cluster:

fs_defaultfs	This is equivalent to `fs.defaultFS` (`fs.default.name`) in Hadoop configuration.
webhdfs_url	You can also set this to be the HttpFS URL. The default value is the HTTP port on the NameNode.
hadoop_hdfs_home	This is the home of your Hadoop HDFS installation. It is the root of the Hadoop untarred directory or usually `/usr/lib/hadoop`.
hadoop_bin	Use this as the HDFS Hadoop launcher script, which is usually `/usr/bin/hadoop`.
hadoop_conf_dir	This is the configuration directory of the HDFS, typically `/etc/hadoop/conf`.

2. Configure YARN (MR2) Cluster.

   Hue supports only one YARN cluster currently.

   Ensure that you define the YARN cluster under the `[[[default]]]` sub-section. Use the following variables to configure the YARN cluster:

resourcemanager_host	The host running the ResourceManager.
resourcemanager_port	The port for the ResourceManager IPC service.
submit_to	Set this property to true. Hue will be submitting jobs to this Yarn cluster. But note that JobBrowser will not be able to show MR2 jobs.
hadoop_mapred_home	This is the home of your Hadoop MapReduce installation. It is the root of HDP Hadoop-MapReduce directory (`/usr/lib/hadoop-mapreduce`). If `submit_to` is `true` for this cluster, this configuration value is set as the `$HADOOP_MAPRED_HOME` for BeeswaxServer and child shell processes.
hadoop_bin	Use this as the Yarn/MR2 Hadoop launcher script (`/usr/bin/hadoop`).
hadoop_conf_dir	This is the configuration directory of the Yarn/MR2 service, typically set to `/etc/hadoop/conf`.

7.4.3. Configure Beeswax

In the `[beeswax]` section of the configuration file, you can optionally specify the following:

beeswax_server_host	The hostname or IP that the Beeswax Server should bind to. By default it binds to localhost, and therefore only serves local IPC clients.
hive_home_dir	The base directory of your Hive installation.
hive_conf_dir	The directory containing your `hive-site.xml` Hive configuration file.
beeswax_server_heapsize	The heap size (`-Xmx`) of the Beeswax Server.

7.4.4. Configure JobDesigner and Oozie

In the `[liboozie]` section of the configuration file, you should specify:

oozie_url	The URL of the Oozie service as specified by the `OOZIE_URL` environment variable for Oozie.

7.4.5. Configure UserAdmin

In the `[useradmin]` section of the configuration file, you can optionally specify:

default_user_group	The name of a default group that is suggested when creating a user manually. If the `LdapBackend` or `PamBackend` are configured for user authentication, new users will automatically be members of the default group.

7.4.6. Configure WebHCat

In the `[HCatalog]` section of the `hue.ini` configuration file, update the following property:

```
templeton_url="http://hostname:50111/templeton/v1/"
```

For example,

```
templeton_url="http://localhost:50111/templeton/v1/"
```

7.5. Start Hue

As a root user, execute the following command on the Hue Server:

```
/etc/init.d/hue start
```

This command starts several subprocesses corresponding to the different Hue components.

Note

To stop Hue, execute the following command:

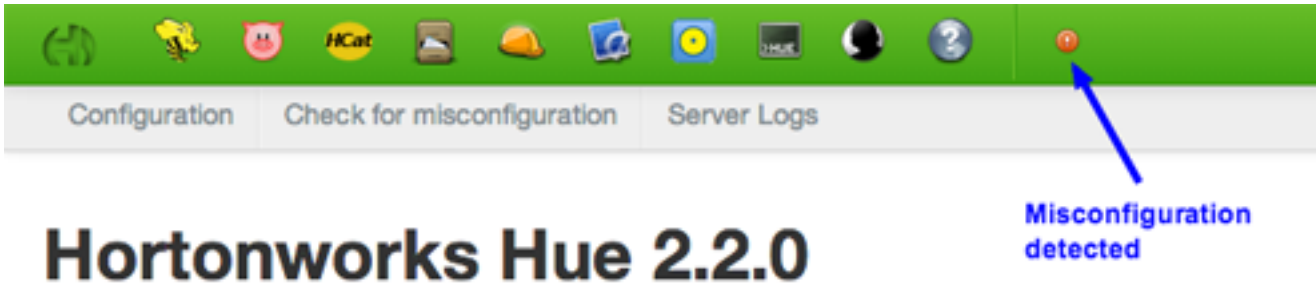
```
/etc/init.d/hue stop
```

To restart Hue, execute the following command:

```
/etc/init.d/hue restart
```

7.6. Validate Configuration

For any invalid configurations, Hue will display a red alert icon on the top navigation bar:



To view the configuration of an existing Hue instance, either browse to `http://myserver:8888/dump_config` or use the **About** menu.

8. Installing WebHCat

This section describes installing and testing WebHCat, which provides a REST interface to Apache HCatalog services like job submission and eventing.

Use the following instructions to install WebHCat:

1. Install the WebHCat RPMs
2. Set Directories and Permissions
3. Modify WebHCat Configuration Files
4. Set Up HDFS User and Prepare WebHCat Directories On HDFS
5. Validate the Installation

8.1. Install the WebHCat RPMs

On the WebHCat server machine, install the necessary RPMs.

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

8.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Pig configuration files :

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute these commands on your WebHCat server machine to create log and pid directories.

   ```
   mkdir -p $WEBHCAT_LOG_DIR
   chown -R $WEBHCAT_USER:$HADOOP_GROUP $WEBHCAT_LOG_DIR
   chmod -R 755 $WEBHCAT_LOG_DIR
   ```

   ```
   mkdir -p $WEBHCAT_PID_DIR
   chown -R $WEBHCAT_USER:$HADOOP_GROUP $WEBHCAT_PID_DIR
   chmod -R 755 $WEBHCAT_PID_DIR
   ```

   where:

   - *$WEBHCAT_LOG_DIR* is the directory to store the WebHCat logs. For example, `var/log/webhcat`.

- *$WEBHCAT_PID_DIR* is the directory to store the WebHCat process ID. For example, `/var/run/webhcat`.
- *$WEBHCAT_USER* is the user owning the WebHCat services. For example, `hcat`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

8.3. Modify WebHCat Configuration Files

Use the following instructions to modify the WebHCat config files:

1. Extract the WebHCat configuration files

   From the downloaded `scripts.zip` file, extract the files in `configuration_files/webhcat` directory to a temporary location.

2. Modify the configuration files

   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. See Define Environment Parameters for more information.

   a. Edit the `webhcat-env.xml` and modify the following properties:

```
<property>
 <name>templeton.hive.properties</name>
 <value>hive.metastore.local=false, hive.metastore.uris=thrift:/
/$metastore.server.full.hostname:9083,hive.metastore.sasl.enabled=no,
hive.metastore.execute.setugi=true</value>
 <description>Properties to set when running Hive.</description>
</property>
```

```
<property>
 <name>templeton.zookeeper.hosts</name>
 <value>$zookeeper1.full.hostname:2181,$zookeeper1.full.hostname:2181,..
</value>
 <description>ZooKeeper servers, as comma separated HOST:PORT pairs.</
description>
</property>
```

3. Set up the WebHCat configuration files.

   a. Delete any existing WebHCat configuration files:

```
rm -rf  $WEBHCAT_CONF_DIR/*
```

   b. Copy all the config files to *$WEBHCAT_CONF_DIR* and set appropriate permissions:

```
chown -R $WEBHCAT_USER:$HADOOP_GROUP $WEBHCAT_CONF_DIR
chmod -R 755 $WEBHCAT_CONF_DIR
```

   where:

- *$WEBHCAT_CONF_DIR* is the directory to store theWebHCat configuration files. For example, `/etc/hcatalog/conf/webhcat`.
- *$WEBHCAT_USER* is the user owning the WebHCat services. For example, `hcat`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

8.4. Set Up HDFS User and Prepare WebHCat Directories On HDFS

1. Set up the WebHCat user.

```
Login as $HDFS_USER
hadoop fs -mkdir /user/$WEBHCAT_USER
hadoop fs -chown -R $WEBHCAT_USER:$HDFS_USER /user/$WEBHCAT_USER
hadoop fs -mkdir /apps/webhcat
```

2. Prepare WebHCat directories on HDFS.

```
hadoop dfs -copyFromLocal /usr/share/HDP-webhcat/pig.tar.gz /apps/webhcat/
hadoop dfs -copyFromLocal /usr/share/HDP-webhcat/hive.tar.gz /apps/webhcat/
hadoop dfs -copyFromLocal /usr/lib/hadoop-mapreduce/hadoop-streaming*.jar /
apps/webhcat/
```

3. Set appropriate permissions for the HDFS user and the webhcat directory.

```
hadoop fs -chown -R $WEBHCAT_USER:users /apps/webhcat
hadoop fs -chmod -R 755 /apps/webhcat
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$WEBHCAT_USER* is the user owning the WebHCat services. For example, `hcat`.

8.5. Validate the Installation

1. Start the WebHCat server.

```
<login as $WEBHCAT_USER>
/usr/lib/hcatalog/sbin/webhcat_server.sh start
```

2. From the browser, type:

```
http://$WebHCat.server.full.hostname:50111/templeton/v1/status
```

You should see the following output:

```
{"status":"ok","version":"v1"}
```

9. Installing HBase and Zookeeper

This section describes installing and testing Apache HBase, a distributed, column-oriented database that provides the ability to access and manipulate data randomly in the context of the large blocks that make up HDFS. It also describes installing and testing Apache ZooKeeper, a centralized tool for providing services to highly distributed systems.

9.1. Install the HBase and ZooKeeper RPMs

In a terminal window, type:

```
yum install zookeeper hbase
```

9.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to create appropriate directories:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute the following commands on all nodes:

```
mkdir -p $HBASE_LOG_DIR;
chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_LOG_DIR;
chmod -R 755 $HBASE_LOG_DIR;
```

```
mkdir -p $HBASE_PID_DIR;
chown -R $HBASE_USER:$HADOOP_GROUP  $HBASE_PID_DIR;
chmod -R 755  $HBASE_PID_DIR;
```

```
mkdir -p $ZOOKEEPER_LOG_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_LOG_DIR;
chmod -R 755 $ZOOKEEPER_LOG_DIR;
```

```
mkdir -p $ZOOKEEPER_PID_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_PID_DIR;
chmod -R 755 $ZOOKEEPER_PID_DIR;
```

```
mkdir -p $ZOOKEEPER_DATA_DIR;
chmod -R 755 $ZOOKEEPER_DATA_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_DATA_DIR
```

   where:

   - *$HBASE_LOG_DIR* is the directory to store the HBase logs. For example, `/var/log/hbase`.

- *$HBASE_PID_DIR* is the directory to store the HBase process ID. For example, `/var/run/hbase`.
- *$HBASE_USER* is the user owning the HBase services. For example, `hbase`.
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.
- *$ZOOKEEPER_LOG_DIR* is the directory to store the ZooKeeper logs. For example, `/var/log/zookeeper`.
- *$ZOOKEEPER_PID_DIR* is the directory to store the ZooKeeper process ID. For example, `/var/run/zookeeper`.
- *$ZOOKEEPER_DATA_DIR* is the directory where ZooKeeper will store data. For example, `/grid/hadoop/zookeeper/data`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

3. Initialize the zookeeper data directories with the 'myid' file. Create one file per Zookeeper server, and put the number of that server in each file.

```
vi $ZOOKEEPER_DATA_DIR/myid
```

In the myid file on the first server, enter the corresponding number:

```
1
```

In the myid file on the second server, enter the corresponding number:

```
2
```

In the myid file on the second server, enter the corresponding number:

```
3
```

9.3. Set Up the Configuration Files

There are several configuration files that need to be set up for HBase and ZooKeeper.

- Extract the HBase and ZooKeeper configuration files.

  From the downloaded `scripts.zip` file, extract the files in `configuration_files/hbase` and `configuration_files/zookeeper` directory to separate temporary directories.

- Modify the configuration files.

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

  1. Edit `zoo.cfg` and modify the following properties:

```
dataDir=$zk.data.directory.path

server.1=$zk.server1.full.hostname:2888:3888

server.2=$zk.server2.full.hostname:2888:3888

server.3=$zk.server3.full.hostname:2888:3888
```

2. Edit the `hbase-site.xml` and modify the following properties:

```
<property>
 <name>hbase.rootdir</name>
 <value>hdfs://$hbase.namenode.full.hostname:8020/apps/hbase/data</value>

 <description>Enter the HBase NameNode server hostname</description>
</property>

<property>
 <name>hbase.master.info.bindAddress</name>
 <value>$hbase.master.full.hostname</value>
 <description>Enter the HBase Master server hostname</description>
</property>

<property>
 <name>hbase.zookeeper.quorum</name>
 <value>$zk.server1.full.hostname,$zk.server2.full.hostname,$zk.server3.
full.hostname</value>
 <description>Comma separated list of Zookeeper servers (match to what is
 specified in zoo.cfg but without portnumbers)</description>
</property>
```

3. Edit the `regionservers` file and list all the RegionServers hostnames (separated by newline character) in your environment. For example, see the sample `regionservers` file with hostnames `RegionServer1` through `RegionServer9`.

```
RegionServer1
RegionServer2
RegionServer3
RegionServer4
RegionServer5
RegionServer6
RegionServer7
RegionServer8
RegionServer9
```

- Copy the configuration files

1. On all hosts create the config directory:

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

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

2. Copy all the HBase configuration files to *$HBASE_CONF_DIR* and the ZooKeeper configuration files to *$ZOOKEEPER_CONF_DIR* directory.

3. Set appropriate permissions:

```
chmod a+x $HBASE_CONF_DIR/;
chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_CONF_DIR/../  ;
chmod -R 755 $HBASE_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`.
- *$ZOOKEEPER_CONF_DIR* is the directory to store the ZooKeeper configuration files. For example, `/etc/zookeeper/conf`.
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.

9.4. Validate the Installation

Use these steps to validate your installation.

1. Start HBase and ZooKeeper.

   a. Execute this command from the each ZooKeeper node:

   ```
   <login as $ZOOKEEPER_USER>
   /usr/lib/zookeeper/bin/zkServer.sh start $ZOOKEEPER_CONF_DIR/zoo.cfg
   ```

   b. Execute this command from the HBase Master node:

   ```
   <login as $HDFS_USER>
   /usr/lib/hadoop/bin/hadoop fs -mkdir /apps/hbase
   /usr/lib/hadoop/bin/hadoop fs -chown -R hbase /apps/hbase
   <login as $HBASE_USER>
   /usr/lib/hbase/bin/hbase-daemon.sh --config $HBASE_CONF_DIR start master
   ```

   c. Execute this command from each HBase Region Server node:

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

   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`.

- *$ZOOKEEPER_CONF_DIR* is the directory to store the ZooKeeper configuration files. For example, `/etc/zookeeper/conf`.
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.

2. Smoke Test HBase and ZooKeeper.

   From a terminal window, enter:

   ```
   su - $HBASE_USER
   hbase shell
   ```

   In the Hbase shell, enter the following commands:

   ```
   echo status
   ```

10. Installing Apache Oozie

This section describes installing and testing Apache Oozie, a server based workflow engine optimized for running workflows that execute Hadoop jobs.

Complete the following instructions to install Oozie:

1. Install the Oozie RPMs
2. Set Directories and Permissions
3. Set Up the Oozie Configuration Files
4. Validate the Installation

10.1. Install the Oozie RPMs

1. On Oozie server, install the necessary RPMs.

```
yum install oozie extjs-2.2-1
```

```
yum install oozie oozie-client
```

2. Add the ExtJS library to the Oozie application.

```
cd /usr/lib/oozie
mkdir libext
cp /usr/share/HDP-oozie/ext-2.2.zip libext/
```

3. Add LZO JAR files.

```
cp /usr/lib/hadoop/lib/hadoop-lzo-0.5.0.jar libext/
```

10.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Oozie configuration files:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute the following commands on your Oozie server:

```
mkdir -p $OOZIE_DATA;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_DATA;
chmod -R 755 $OOZIE_DATA;
```

```
mkdir -p $OOZIE_LOG_DIR;
```

```
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_LOG_DIR;
chmod -R 755 $OOZIE_LOG_DIR;
```

```
mkdir -p $OOZIE_PID_DIR;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_PID_DIR;
chmod -R 755 $OOZIE_PID_DIR;
```

```
mkdir -p $OOZIE_TMP_DIR;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_TMP_DIR;
chmod -R 755 $OOZIE_TMP_DIR;
```

where:

- *$OOZIE_DATA* is the directory to store the Oozie data. For example, `/var/db/oozie`.
- *$OOZIE_LOG_DIR* is the directory to store the Oozie logs. For example, `/var/log/oozie`.
- *$OOZIE_PID_DIR* is the directory to store the Oozie process ID. For example, `/var/run/oozie`.
- *$OOZIE_TMP_DIR* is the directory to store the Oozie temporary files. For example, `/var/tmp/oozie`.
- *$OOZIE_USER* is the user owning the Oozie services. For example, `oozie`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

10.3. Set Up the Oozie Configuration Files

Complete the following instructions to set up Oozie configuration files:

1. Extract the Oozie configuration files.

   From tthe downloaded `scripts.zip` file, extract the files from the `configuration_files/oozie` directory to a temporary directory.

2. Modify the configuration files.

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

   a. Edit the `oozie-site.xml` and modify the following properties:

```
<property>
 <name>oozie.base.url</name>
 <value>http://$oozie.full.hostname:11000/oozie</value>
 <description>Enter your Oozie server hostname.</description>
</property>
```

```
<property>
 <name>oozie.service.StoreService.jdbc.url</name>
 <value>jdbc:derby:$OOZIE_DATA_DIR/$soozie.db.schema.name-db;create=
true</value>

</property>
```

```
<property>
 <name>oozie.service.JPAService.jdbc.driver</name>
 <value>org.apache.derby.jdbc.EmbeddedDriver</value>
</property>

<property>
 <name>oozie.service.JPAService.jdbc.username</name>
 <value>$OOZIE_DBUSER</value>
</property>

<property>
 <name>oozie.service.JPAService.jdbc.password</name>
 <value>$OOZIE_DBPASSWD</value>
</property>
```

b. Edit the `oozie-env.sh` and modify the following properties to match the directories created:

```
<property>
 <name>OOZIE_LOG_DIR</name>
 <value>/var/log/oozie</value>
 <description>Use value from $OOZIE_LOG_DIR </description>
</property>

<property>
 <name>OOZIE_PID_DIR</name>
 <value>/var/run/oozie</value>
 <description>Use value from $OOZIE_PID_DIR </description>
</property>

<property>
 <name>OOZIE_DATA_DIR</name>
 <value>/var/db/oozie</value>
 <description>Use value from $OOZIE_DATA_DIR </description>
</property>
```

3. Copy the Configuration Files

   On your Oozie server create the config directory, copy the config files and set the permissions:

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

4. Copy all the config files to *$OOZIE_CONF_DIR* directory.

5. Set appropriate permissions.

```
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_CONF_DIR/../ ;
chmod -R 755 $OOZIE_CONF_DIR/../ ;
```

where:

- *$OOZIE_CONF_DIR* is the directory to store Oozie configuration files. For example, `/etc/oozie/conf`.
- *$OOZIE_DATA* is the directory to store the Oozie data. For example, `/var/db/oozie`.

- *$OOZIE_LOG_DIR* is the directory to store the Oozie logs. For example, `/var/log/oozie`.
- *$OOZIE_PID_DIR* is the directory to store the Oozie process ID. For example, `/var/run/oozie`.
- *$OOZIE_TMP_DIR* is the directory to store the Oozie temporary files. For example, `/var/tmp/oozie`.
- *$OOZIE_USER* is the user owning the Oozie services. For example, `oozie`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

10.4. Validate the Installation

Use these steps to validate your installation.

1. Run the setup script to prepare the Oozie Server

```
cd /usr/lib/oozie/
bin/oozie-setup.sh prepare-war
```

2. Create the Oozie DB schema

```
cd /usr/lib/oozie/
bin/ooziedb.sh create -sqlfile oozie.sql -run Validate DB Connection
```

3. Start the Oozie server:

```
<login as $oozie_user>
cd /usr/lib/oozie/
/usr/lib/oozie/bin/oozie-start.sh
```

4. Confirm that you can browse to the Oozie server:

```
http://{oozie.full.hostname}:11000/oozie
```

5. Access the Oozie Server with the Oozie client.

```
oozie admin -oozie http://$oozie.full.hostname:11000/oozie -status
```

You should see the following output:

```
System mode: NORMAL
```

11. Installing Apache Sqoop

This section describes installing and testing Apache Sqoop, a component that provides a mechanism for moving data between HDFS and external structured datastores. Use the following instructions to deploy Apache Sqoop:

1. Install the Sqoop RPMs
2. Set Up the Sqoop Configuration
3. Install the Sqoop RPMs

11.1. Install the Sqoop RPMs

On all nodes where you plan to use the Sqoop client, install the following RPMs:

```
yum install sqoop
```

11.2. Set Up the Sqoop Configuration

This section describes how to set up and edit the deployment configuration files for Sqoop.

Use the following instructions to set up Sqoop configuration files:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. From the downloaded `scripts.zip` file, extract the files from the `configuration_files/sqoop` directory to a temporary directory.

3. Modify the configuration files.

   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. See Define Environment Parameters for more information.

   a. From the file you downloaded in Download Companion Files extract the files in `configuration_files/sqoop` to a temporary directory.

   b. Copy all the config files to the

   ```
   <copy the config files to $SQOOP_CONF_DIR >
   ```

   where *$SQOOP_CONF_DIR* is he directory to store the Sqoop configuration files. For example, `/usr/lib/sqoop/conf`.

11.3. Validate the Installation

Execute the following command. You should see the Sqoop version information displayed.

```
sqoop version | grep 'Sqoop [0-9].*'
```

12. Installing and Configuring Flume in HDP

You can manually install and configure Apache Flume to work with the Hortonworks Data Platform (HDP).

Use the following links to install and configure Flume for HDP:

12.1. Understand Flume

Flume is a top-level project at the Apache Software Foundation. While it can function as a general-purpose event queue manager, in the context of Hadoop it is most often used as a log aggregator, collecting log data from many diverse sources and moving them to a centralized data store.

Note

What follows is a very high-level description of the mechanism. For more information, access the Flume HTML documentation set installed with Flume. After you install Flume, access the documentation set at `file:///usr/lib/flume/docs/index.html` on the host on which Flume is installed. The *"Flume User Guide"* is available at `file:///usr/lib/flume/docs/FlumeUserGuide.html`. If you have access to the Internet, the same documentation is also available at the Flume website, flume.apache.org.

12.1.1. Flume Components

A Flume data flow is made up of five main components: Events, Sources, Channels, Sinks, and Agents.

Events — An event is the basic unit of data that is moved using Flume. It is similar to a message in JMS and is generally small. It is made up of headers and a byte-array body.

Sources — The source receives the event from some external entity and stores it in a channel. The source must understand the type of event that is sent to it: an Avro event requires an Avro source.

Channels — A channel is an internal passive store with certain specific characteristics. An in-memory channel, for example, can move events very quickly, but does not provide persistence. A file based channel provides persistence. A source stores an event in the channel where it stays until it is consumed by a sink. This temporary storage lets source and sink run asynchronously.

Sinks — The sink removes the event from the channel and forwards it on either to a destination, like HDFS, or to another agent/dataflow. The sink must output an event that is appropriate to the destination.

Agents — An agent is the container for a Flume data flow. It is any physical JVM running Flume. The same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

12.2. Install Flume

Flume is included in the HDP repository, but it is not installed automatically as part of the standard HDP installation process.

12.2.1. Prerequisites

1. At least core Hadoop installed on your system. See HDP Deployment Options for more information.
2. You must have set up your JAVA_HOME environment variable per your operating system. See here for instructions on installing JDK.

12.2.2. Installation

To install Flume, from a terminal window type:

- For RHEL or CentOS

```
yum install flume
yum install flume-node #This installs init scripts
```

- For SLES

```
zypper install flume
zypper install flume-node #This installs init scripts
```

or

12.2.3. Users

The installation process automatically sets up the appropriate `flume` user and `flume` group in the operating system.

12.2.4. Directories

The main Flume files are located in `/usr/lib/flume` and the main configuration files are located in `/etc/flume/conf`.

12.3. Configure Flume

You configure Flume by using a properties file, which is specified on Flume start-up. The init scripts installed by `flume-node` bring up a single Flume agent on any host, using the contents of `/etc/flume/conf/flume-conf`.

To see what configuration properties you can adjust, a template for this file is installed in the configuration directory at: `/etc/flume/conf/flume-conf.properties.template`. A second template file exists for setting environment variables automatically at start-up: `/etc/flume/conf/flume-env.sh.template`.

Common configuration option choices include the following:

- Set primary configuration options in `/etc/flume/conf/flume-conf`:
  - If you are using the HDFS sink make sure the target folder is in HDFS
- Set environment options in `/etc/flume/conf/flume-env.sh`:
  - To enable JMX monitoring, add the following properties to JAVA_OPTS

```
JAVA_OPTS="-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=4159
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false"
```

  - To enable Ganglia monitoring, add the following properties to JAVA_OPTS

```
JAVA_OPTS="-Dflume.monitoring.type=ganglia
-Dflume.monitoring.hosts=<ganglia-server>:8660"
```

    Where <ganglia-server> is the name of the Ganglia server host.

  - To optimize the heap size, add the following properties to JAVA_OPTS

```
JAVA_OPTS= "-Xms100m -Xmx200m"
```

- Set the log directory for log4j in `/etc/flume/conf/log4j.properties`

```
flume.log.dir=/var/log/flume
```

12.4. Start Flume

There are two options for starting Flume.

- Start Flume directly. On the Flume host:

```
/etc/rc.d/init.d/flume-node start
```

- Start Flume as a service. On the Flume host:

```
service flume-node start
```

12.5. HDP and Flume

Flume ships with many source, channel, and sink types. For use with HDP the following types have been thoroughly tested:

12.5.1. Sources

- Exec (basic, restart)
- Syslogtcp
- Syslogudp

12.5.2. Channels

- Memory
- File

12.5.3. Sinks

- HDFS: secure, nonsecure
- HBase

12.6. A Simple Example

The following snippet shows some of the kinds of properties that can be set using the properties file. For more detailed information, see the *"Flume User Guide"*.

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory
agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = tail -f /etc/passwd
agent.sinks = hdfsSink
agent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://hdp/user/root/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

The source here is defined as an `exec` source, the agent runs a given command on start-up which streams data to `stdout`, where the source gets it. In this case, the command is a Python test script. The channel is defined as an in-memory channel and the sink is an HDFS sink.

13. Manual Install Appendix: Tarballs

Individual links to the Apache structured tarball files for the projects included with Hortonworks Data Platform are listed below:

- **RHEL 5 and CentOS 5: [Not supported in Beta]**
- **RHEL 6 and CentOS 6:**

Table 13.1. RHEL/CentOS 6

Project	Download
Hadoop	hadoop-2.1.0.2.0.5.0-67.tar.gz
Pig	pig-0.11.2.2.0.5.0-67.tar.gz
Hive and HCatalog	hive-0.11.0.2.0.5.0-67.tar.gz hcatalog-0.11.0.2.0.5.0-67.tar.gz
HBase and ZooKeeper	hbase-0.95.2.2.0.5.0-67-bin.tar.gz zookeeper-3.4.5.2.0.5.0-67.tar.gz
Oozie	oozie-4.0.0.2.0.5.0-67-distro.tar.gz
Sqoop	sqoop-1.4.4.2.0.5.0-67.bin__hadoop-2.1.0.2.0.5.0-67.tar.gz
Flume	apache-flume-1.4.0.2.0.5.0-67-bin.tar.gz

14. Uninstalling HDP

Use the following instructions to uninstall HDP:

1. Stop all the installed HDP services.
2. If HCatalog is installed, execute the following command on all the cluster nodes:

```
yum remove hcatalog\*
```

3. If Hive is installed, execute the following command on all the cluster nodes:

```
yum remove hive\*
```

4. If Tez is installed, execute the following command on all the cluster nodes:

```
yum remove tez
```

5. If HBase is installed, execute the following command on all the cluster nodes:

```
yum remove hbase\*
```

6. If ZooKeeper is installed, execute the following command on all the cluster nodes:

```
yum remove zookeeper\*
```

7. If Pig is installed, execute the following command on all the cluster nodes:

```
yum remove pig\*
```

8. If compression libraries are installed, execute the following command on all the cluster nodes:

```
yum remove snappy\*
yum remove hadoop-lzo\*
```

9. Uninstall Hadoop. Execute the following command on all the cluster nodes:

```
yum remove hadoop\*
```

10. Uninstall ExtJS libraries and MySQL connector. Execute the following command on all the cluster nodes:

```
yum remove extjs-2.2-1 mysql-connector-java-5.0.8-1\*
```

11. Delete Hadoop directories. If your Hadoop Home directory is "/usr/lib/hadoop" for example, delete that directory and its subdirectories.

```
rm -rf /usr/lib/hadoop
```