Pivotal™

Product Documentation

Pivotal™ HD

Version 2.1

Installation and Administrator Guide

Rev: A03

Notice

Copyright

Pivotal Software, Inc. believes the information in this publication is accurate as of its publication date. The information is subject to change without notice. THE INFORMATION IN THIS PUBLICATION IS PROVIDED "AS IS." PIVOTAL SOFTWARE, INC. ("Pivotal") MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use of Open Source

This product may be distributed with open source code, licensed to you in accordance with the applicable open source license. If you would like a copy of any such source code, Pivotal will provide a copy of the source code that is required to be made available in accordance with the applicable open source license. Pivotal may charge reasonable shipping and handling charges for such distribution.

About Pivotal Software, Inc.

Greenplum transitioned to a new corporate identity (Pivotal, Inc.) in 2013. As a result of this transition, there will be some legacy instances of our former corporate identity (Greenplum) appearing in our products and documentation. If you have any questions or concerns, please do not hesitate to contact us through our web site: *http://support.pivotal.io*.

Published September 2014

Updated November 2014

Contents

Chapter 1

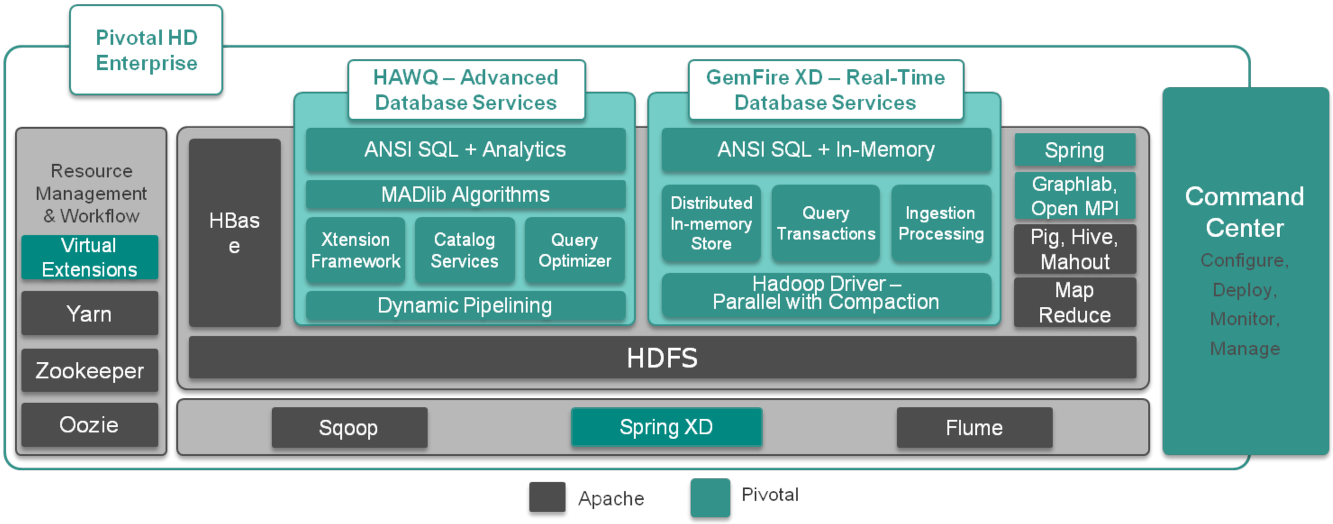
Overview of PHD

Pivotal HD Enterprise is an enterprise-capable, commercially supported distribution of Apache Hadoop packages targeted to traditional Hadoop deployments.

PHD Architecture

Pivotal HD Enterprise is a commercially-supported distribution of the Apache Hadoop stack. The figure below displays how each Apache and Pivotal component fits into the overall architecture of Pivotal HD Enterprise:

Pivotal HD Architecture



Pivotal HD Enterprise includes the following Apache and Pivotal components:

- **Core Apache Stack:**
  - Hadoop
    - HDFS
    - YARN
  - Zookeeper
  - HBase
  - Hive
  - Pig
  - Mahout
  - Flume
  - Sqoop
  - Oozie

Pivotal HD Enterprise enriches the Apache stack distribution by providing the following:

- **Advanced Database Services (ADS)**
  - **HAWQ** - HAWQ adds SQL's expressive power to Hadoop. By adding rich, proven parallel SQL processing facilities, HAWQ renders queries faster than any other Hadoop-based query interface.
  - **PXF** - Extensibility layer to provide support for external data formats such as HBase and Hive.
- **Pivotal Command Center (PCC)** - PCC Is a Web-based interface for configuration and deployment of clusters, and for monitoring & management of a PHD environment. With the help of PCC, system administrators can determine if the PHD cluster is running efficiently, quickly diagnose functional or performance issues, and performs cluster management tasks when required. PCC includes a CLI (command line interface) and a GUI. You can deploy and configure most of the Hadoop services, as well as HAWQ, and PXF, using either the CLI or the GUI (See *Deployment Options* on page 16). You can start and stop the clusters using either the CLI or the GUI.

Note: This documentation covers operations performed via the CLI. For Pivotal Command Center GUI operations; including configuring and deploying clusters, see the *Pivotal Command Center 2.x User Guide*.

PCC stores the metadata for Hadoop cluster nodes and services, the cluster configuration and the system metrics in a PostgreSQL database.

- **Pivotal Real Time Services (PRTS)**- Pivotal HD 2.x includes support for GemFire XD (GFXD), an offering of PRTS.
- **Hamster** - Developed by Pivotal, Hamster is a framework that enables users to run MPI programs on Apache Hadoop YARN platform. (OpenMPI is a A High Performance Message Passing Library.)
- **GraphLab** - GraphLab is a powerful new system for designing and implementing parallel algorithms in machine learning. It is a graph-based, high performance, distributed computation framework written in C++ that makes use of MPI and has its own programming model.

About Supported Pivotal HD Services

The following services can be deployed and configured via the Pivotal Command Center CLI, or manually.

- HDFS
- YARN
- ZooKeeper
- Hbase
- Hive
- HAWQ
- PXF
- GemFire XD
- Pig
- Mahout

The following services can only be deployed and configured manually (see the *Stack and Tools Reference* for details)

- Flume
- Sqoop
- Oozie
- Hamster
- GraphLab

HDFS

HDFS is a fault tolerant distributed file system which is designed to run on commodity hardware.

The following table shows HDFS service roles:

Role Name	Description
NameNode	The NameNode serves as both directory namespace manager and "inode table" for the Hadoop File System (HDFS). Each HDFS deployment must have a running NameNode.
Secondary NameNode	The Secondary NameNode periodically downloads the current NameNode image and edits log files. It joins them into a new image and uploads the new image back to the primary NameNode.
DataNodes	A DataNode stores data in the HDFS. A functional filesystem has more than one DataNode, with data replicated across all nodes.

Role Name	Description
Hadoop Client	A client machine has Hadoop installed with all the cluster settings, but is not a Master or Slave. Instead, the role of the client is to load data into the cluster, submit Map Reduce jobs that describe how to process the data, and then retrieve or view the results of the finished job.
Journalnodes *	A group of daemons to maintain the namenode edits information. These are used by both active and standby namenodes in a HA enabled cluster to keep their state synchronized.
Standby Namenode *	Namenode running on a different host in standby mode in a HA enabled cluster. This will take over as the active namenode if the current active namenode fails.

*Only applicable for HA enabled clusters.

YARN

YARN is a framework that facilitates writing distributed processing frameworks and applications and supports MapReduce version 2.

The following table shows YARN service roles:

Role Name	Description
Resource Manager	The ResourceManager is the master that manages all the cluster resources running on the YARN system.
Node Manager	The NodeManager manages resources on a particular node.
History Server	The History Server stores a history of the mapreduce jobs run on the cluster.

ZooKeeper

Zookeeper is a centralized service that enable distributed synchronization and manages configuration across a cluster.

The following table shows ZooKeeper service roles:

Role Name	Description
Zookeeper Server	ZooKeeper Quorum Servers

HBase

HBase is a distributed, column-oriented database that uses HDFS for storing data.

The following table shows HBase service roles:

Role Name	Description
HBase Master	The Master server is responsible for monitoring all RegionServer instances in the cluster, and is the interface for all metadata changes.
HBase RegionServer	It is responsible for serving and managing regions which typically coexist with datanodes.
HBase Client	It is responsible for accessing HBase service.

Note:

- HBase requires that you have installed HDFS, YARN, and Zookeeper.
- Pivotal HD installs ZooKeeper if you have not installed it.
- HBase does not manage the Zookeeper service.

Hive

Hive is a *data warehouse* infrastructure that provides an interface similar to SQL on top of Hadoop.

Role Name	Description
Hive Metastore	The metastore stores the metadata for all Hive tables and partitions. Postgres database is used as the datastore
Hive Server	Also known as thrift server, is used by clients written in Java, C++ etc to access Hive
Hive Client	This is a launcher or gateway node which is used to launch hive jobs

Note: Hive requires HDFS and YARN.

HAWQ

HAWQ is a parallel SQL query engine that marries the Pivotal Analytic Database and Hadoop 2.0 and is optimized for analytics, with full transaction support. The following table shows HAWQ service roles:

Role Name	Description
HAWQ Master	Stores the top-level metadata, as well as building the query plan
HAWQ StandbyMaster	This is a standby for the HAWQ Master
HAWQ Segments	Manages a shard of each table which typically coexist with datanodes

Note: HAWQ requires HDFS.

PXF

PXF is an extended framework that combines the Pivotal Analytic Database engine (HAWQ) with enterprise class Apache Hadoop, HBase and Hive. The PXF service runs as a java agent on existing Hadoop, HBase and Hive nodes and enables HAWQ to consume data created by the external services.

Note: PXF requires HDFS and HAWQ.

If you do not install PXF via the CLI, and choose to install it later, refer to the *HAWQ 1.2 Administrator Guide* for details.

GemFire XD

GemFire XD is a memory-optimized, distributed data store that is designed for applications that have demanding scalability and availability requirements.

Note that you cannot start GemFire XD (`gfxd`) using the `icm_client start` command. See *http://gemfirexd.docs.pivotal.io/latest/userguide/index.html#getting_started/topics/install_platform.html* in the GemFire XD documentation for information about how to configure and start GemFire XD members.

Service Roles/Ports

The following table shows GemFire service roles:

Role Name	Description	Port
gfxd-locator	The GemFire XD locator process provides discovery services for all members in a GemFire XD distributed system. A locator also provides load balancing and failover for thin client connections. As a best practice, deploy a locator in its own process (LOCATOR=local_only) to support network partitioning detection.	1527
gfxd-server	A GemFire XD server hosts database schemas and provides network connectivity to other GemFire XD members and clients. You can deploy additional servers as necessary to increase the capacity for in-memory tables and/or provide redundancy for your data.	1527

Pig

Pig is a data flow language used in the analysis of large data sets using mapreduce.

Role Name	Description
Pig Client	This is a launcher or gateway node which is used to launch Pig jobs

Note: Pig requires HDFS and YARN/MapReduce.

Mahout

Mahout provides a collection of distributed *machine learning* algorithms on *Hadoop*.

Role Name	Description
Mahout Client	This is a launcher or gateway node which is used to launch Mahout jobs

Note: Mahout requires HDFS and YARN/MapReduce.

Flume

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

Role Name	Description
Flume Agent	Provide Flume service for generating, processing, and delivering data
Flume Client	This is a launcher or gateway node which is used to launch Flume jobs

Note: Flume requires HDFS and YARN/MapReduce.

Sqoop

Sqoop is a tool designed for efficiently transferring bulk data between *Apache Hadoop*and structured datastores such as relational databases.

Role Name	Description
Sqoop Metastore	Provide shared metadata repository for Sqoop
Sqoop Client	This is a launcher or gateway node which is used to launch sqoop jobs

Note: Sqoop requires HDFS, YARN/MapReduce, and HBase.

Oozie

Oozie is a workflow scheduler system to manage Apache Hadoop jobs.

Role Name	Description
Oozie Metastore	provide Oozie service
Oozie Client	This is a launcher or gateway node which is used to launch Oozie jobs

Note: Oozie requires HDFS, YARN/MapReduce , Pig (optional) and Hive (optional).

Hamster

Hamster is a framework that enables users to run MPI programs on Apache Hadoop YARN platform.

GraphLab

GraphLab is a powerful new system for designing and implementing parallel algorithms in machine learning. It is a graph-based, high performance, distributed computation framework written in C++ that makes use of MPI and has its own programming model.

Chapter 2

Installation Overview

This section provides an overview of the Pivotal HD installation process, along with some recommended best practices.

Command Line Installation Features

Using Pivotal Command Center's CLI to install Pivotal HD provides the following functionality:

Feature	Support
Checking prerequisites	Checks that specified hosts meet the prerequisites to install the supported components.
Supported cluster services	• Installs and configures Hadoop, YARN, ZooKeeper, HBase, Mahout, HAWQ, PXF, Hive, and Pig with default settings. • Reconfigures the supported cluster services. • Multi-cluster support. • Monitors clusters with Pivotal Command Center.
Starting and stopping	• Starts and stops the cluster or individual services. • Ensures that all dependent services start and stop in the correct order.
Logging	Provides installation data logs.
Uninstallation	Uninstalls individual services and Pivotal HD Enterprise.

Related:

Deployment Options on page 16

Planning your PHD Cluster Deployment on page 17

Best Practices for Selecting Hardware on page 18

Best Practices for Deploying Hadoop Services on page 19

Best Practices for High Availability on page 126

Deployment Options

The following table illustrates the deployment options and limitations:

Component		CLI Install	Manual Install (via RPM)
Pivotal Command Center (installs the CLI)			✓
Hadoop MR2: HDFS, YARN		✓	✓
Pig		✓	✓
Hive		✓	✓
HBase		✓	✓
Mahout		✓	✓
Zookeeper		✓	✓
Flume			✓
Sqoop			✓
Oozie			✓
Hamster			✓
GraphLab			✓
Advanced Database Services:	HAWQ	✓	✓
	PXF	✓	✓

Related:

Command Line Installation Features on page 16

Planning your PHD Cluster Deployment on page 17

Best Practices for Selecting Hardware on page 18

Best Practices for Deploying Hadoop Services on page 19

Best Practices for High Availability on page 126

Planning your PHD Cluster Deployment

Before deploying a Hadoop cluster, Pivotal recommends that you consider the following:

- Select the appropriate hardware configuration for your Admin and cluster nodes.
- Map Hadoop services roles to cluster nodes.
- Configure the roles to effectively leverage the underlying hardware platform.
- Determine the number of NameNodes and JournalNodes to use for a High Availability (HA) cluster.

For more information, see:

Best Practices for Selecting Hardware on page 18

Best Practices for Deploying Hadoop Services on page 19

Best Practices for High Availability on page 126

Best Practices for Selecting Hardware

Typically, you should select your cluster node hardware based on the resource requirements of your analytics workload and overall need for data storage. It is hard to anticipate the workload that may run on the cluster, so designing for a specific type of workload could lead to under utilization of hardware resources. Pivotal recommends that you select the hardware for a balanced workload across different types of system resources, but also have the ability to provision more specific resources such as CPU, I/O bandwidth, and Memory, as workload evolves over the time and the demands for it.

Hardware and capacity requirements for cluster nodes can vary depending upon what service roles running on them. Typically, failure of cluster slave nodes is tolerated by PHD services, but disruption to master node can cause service availability issues. Thus, it is important to provide more reliable hardware for master nodes (such as NameNode, YARN Resource manager, HAWQ master) for higher cluster availability.

Overall, when choosing the hardware for cluster nodes, select equipment that lowers power consumption.

Note: Following are not minimum requirements, they are Pivotal best practices recommendations.

Any configuration higher than the minimum recommendations is always preferable.

Cluster Slaves

Cluster slave nodes run Hadoop service slaves such as the Datanode, NodeManager, RegionServer, and SegmentServer.

- 2 CPUs (4 to 8 cores)--- You can also have a single CPU with more (6 to 8) cores and the ability to add additional CPUs, if needed in future. An algorithm to measure this is as follows: total map+reduce tasks per node are ~= 1.5 times number of cores per node. Note: You might consider decreasing the number of map/reduce tasks per node when using PHD with HAWQ and assigning more cores to HAWQ segment servers, based on mixed workload of HAWQ vs. MapReduce.
- 24 to 64GB RAM per node — Typically 1 GB for each Hadoop daemon, such as DataNode, NodeManager, Zookeeper etc., 2 to 3GB for OS and other services; and 1.5 or 2GB for each map/reduce task. **Note**: memory per map/reduce tasks on slave nodes depends on application requirements.
- 4 to 10, 2TB or 3TB disks, 7.2K RPM, SATA drives (JBOD) -- More disks per node provides more I/O bandwidth, although more disk capacity per node could put more memory requirements on the HDFS Namenode. The reason for this is that the total HDFS storage capacity grows with the number of cluster nodes, while average HDFS file size stays small.
- 2 x 2TB or 3TB disks, RAID 1 configured for System OS. It can also store Hadoop daemon logs.
- 1GbE or 10GbE network connectivity within RACK

Cluster Masters

Cluster master nodes run Hadoop service masters such as the NameNode, ResourceManager, and HAWQ Master

You must select more reliable hardware for cluster master nodes.

- Memory (RAM) requirements are higher, depending on the size of the cluster, number of HDFS storage, and number of files. Typical memory ranges would be 24GB to 64 GB.
- Local disk storage requirement is 1 to 2TB, SAS disks, with RAID5/6

Note: Master nodes require less storage than cluster slave nodes.

Pivotal HD Admin Node

Ensure that the Admin node is separate from the cluster nodes, especially if the cluster has more than 15 - 20 nodes. The minimum hardware requirements are as follows:

- 1 Quad code CPU,
- 4 to 8GB RAM,
- 2x2TB SATA disks,
- 1GbE network connectivity

Related:

Best Practices for Deploying Hadoop Services on page 19

Best Practices for High Availability on page 126

Best Practices for Deploying Hadoop Services

When creating your test environment, you can deploy all the Hadoop services and roles on a single node. A test cluster usually comprises 3 to 5 nodes. However, when deploying a production cluster with more nodes, use the following guidelines for better performance, availability, and use:

- Hadoop services Master roles: For example, HDFS NameNode, YARN ResourceManager and History Server, HBase Master, HAWQ Master. These should reside on separate nodes. These services and roles require dedicated resources, since they communicate directly with Hadoop client applications. Running Hadoop slave/application tasks (map/reduce tasks) on the same node interferes with master resource requirements.
- Hadoop services slave roles: For example, HDFS DataNode, YARN NodeManager, HBase RegionServer, HAWQ SegmentServer. These should reside on the cluster slave nodes. This helps provide optimal data access as well as better hardware use.
- HBase requires Zookeeper: Zookeeper should have an odd number of Zookeeper servers. This application does not need dedicated nodes and can reside on the master server with ~ 1GB RAM and a dedicated disk with ~ 1 TB of space.
- Hadoop Clients: For example, Hive, Pig etc. These should be installed on the separate gateway nodes, depending on multi-user application requirements.

At this point you should have numerous systems with defined roles (admin node, namenode, HAWQ master, etc), all ready for installation/deployment of the PHD software distribution.

Related:

Best Practices for Selecting Hardware on page 18

Best Practices for High Availability on page 126

Chapter 3

PHD Pre-Install

This section provides information you'll need, as well as tasks that must be completed, before you install PHD.

Before You Begin Installing PHD

Before you begin your installation, be sure to read the **PHD Release Notes** for information about the latest features, improvements, resolved and known issues; as well as the latest versioning and compatibility information.

We recommend you have a working knowledge of the following:

- **Yum**: Yum enables you to install or update software from the command line. See *http://yum.baseurl.org/*.
- **RPM** (Redhat Package Manager). See information on RPM at **Managing RPM-Based Systems with Kickstart and Yum**. See *http://shop.oreilly.com/product/9780596513825.do?sortby=publicationDate*
- **NTP**. See information on NTP at: *http://www.ntp.org/*
- **SSH** (Secure Shell Protocol). See information on SSH at *http://www.linuxproblem.org/art_9.html*

Note: Refer to the *PHD Pre-Install Checklist* on page 20.

PHD Pre-Install Checklist

The following tasks need to be completed before you begin your PHD installation.

Each task is explained in more detail in subsequent sections; click the task name to jump to those sections.

Step	Task	Description	Completed

1	*DNS Lookup*	Verify that hosts can reach each other using hostnames and IP addresses:
		`# ping -c 3 myhost.mycompany.com  // The return code should be 0`
		`# ping -c  3 192.168.1.2          // The return code should be 0`	

2	*JAVA JDK*	Ensure you're running Oracle Java JDK Version 1.7 on the Admin node. Java version 7 is required; version JDK 1.7u45 is recommended. As `root`, run: `# /usr/java/default/bin/java -version`	

Step	Task	Description	Completed
		If not, download and install the appropriate version from Oracle.	
3	*Verify Package Accessibility*	Verify that all hosts have yum access to an EPEL yum repository. `# yum list <LIST_OF_PACKAGES>` See *PHD Pre-Inst 3 - Verify Package Accessibility* on page 23 for more details and a list of packages. Note that this is not required if the required RPMs are accessible locally.	

4	*Turn Off iptables*	As `root`, run:
		`# chkconfig iptables off`
		`# service iptables stop`
		`# service iptables status`
		`iptables: Firewall is not running.`	

5	*Disable SELinux*	As `root`, run: `# echo 0 > /selinux/enforce`	

Additional Tasks:

Task	Description
sudo Configuration Files on page 25	If you don't use the automatically-created sudo configuration file, you need to manually add some settings to your own sudo configuration file.
Fully Qualified Domain Names (FQDN) on page 26	Make sure that your hostnames are fully qualified domain names (FQDN).
EPEL Yum Repository on page 26	The PHD install expects the required packages to be pre-installed on each host, depending on the software that gets deployed on a particular host. To facilitate install, we recommend that each host have yum access to an EPEL yum repository.

PHD Pre-Inst 1 - DNS Lookup

Before you can begin your PHD installation, verify the following:

Verify that the admin host (the host on which you will be installing PCC) is able to reach every host that will be part of your cluster using its hostname and IP address. We also recommend that every cluster node is able to reach every other cluster node using its hostname and IP address:

```
# ping -c 3 myhost.mycompany.com // The return code should be 0
# ping -c 3 192.168.1.2 // The return code should be 0
```

Next Task:

PHD Pre-Inst 2 - JAVA JDK on page 22

PHD Pre-Inst 2 - JAVA JDK

Before you begin your installation, ensure that you are running Oracle JAVA JDK version 1.7 on the Admin node and that you are not running OpenJDK as your default JDK.

Note: Version 1.7 is required; version 1.7u45 is recommended.

Verify JDK Version

Perform the following steps on the Admin node as both `root` and `gpadmin` users:

```
$ /usr/java/default/bin/java -version
```

The output of this command should contain 1.7 (version number) and JavaHotSpot(TM) (Java version). For example:

```
java version "1.7.0_45"
Java(TM) SE Runtime Environment (build 1.7.0_45-b18)
Java HotSpot(TM) 64-Bit Server VM (build 24.45-b08, mixed mode)
```

If you are not running the correct JDK, download a supported version from the Oracle site at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

Note: If you have manually installed UnlimitedJCEPolicy files prior to upgrading your JDK, you will need to re-install them post upgrade.

Install the JDK on the admin node and add it to alternatives as follows:

```
# /usr/sbin/alternatives --install "/usr/bin/java" "java" "/usr/java/
jdk1.7.0_xx/bin/java" 3
# /usr/sbin/alternatives --install "/usr/bin/javac" "javac" "/usr/java/
jdk1.7.0_xx/bin/javac" 3
# /usr/sbin/alternatives --config java
```

OpenJDK

Make sure you are not running OpenJDK as your default JDK.

If you are running OpenJDK, we recommend you remove it.

To check for all versions of JDK that are running on your system, as `root` run:

```
yum list installed | grep jdk
```

An example output from this command is:

```
java-1.6.0-openjdk.x86_64
java-1.7.0-openjdk.x86_64
jdk.x86_64              2000:1.7.0_45-fcs
```

This indicates that there are three versions of JDK installed, two of them are OpenJDK.

To remove all OpenJDK versions, as `root`, run:

```
yum erase *openjdk*
```

Next Task:

PHD Pre-Inst 3 - Verify Package Accessibility on page 23

PHD Pre-Inst 3 - Verify Package Accessibility

Verify that all packages are available in a local yum repository or that you have yum access to an EPEL yum repository.

Pivotal Command Center and Pivotal HD Enterprise expect some prerequisite packages to be pre-installed on each host, depending on the software that gets deployed on a particular host. In order to have a smoother installation, it is recommended that each host have yum access to an EPEL yum repository. If you have access to the Internet, you can configure your hosts to have access to the external EPEL repositories. However, if your hosts do not have Internet access (or you are deploying onto a large cluster), then having a local yum EPEL repo is highly recommended. This will also give you some control on the package versions you want to deploy on your cluster. See *EPEL Yum Repository* on page 26, for instructions on how to setup a local yum repository or point your hosts to an EPEL repository.

The following packages need to be either already installed on the admin host or be on an accessible yum repository:

- httpd
- mod_ssl
- postgresql
- postgresql-devel
- postgresql-server
- postgresql-jdbc
- compat-readline5
- createrepo
- sigar
- sudo
- python-ldap
- openldap
- openldap-clients
- openldap-servers
- pam_krb5
- sssd
- authconfig
- krb5-workstation
- krb5-libs
- krb5-server

Run the following command on the admin node to make sure that you are able to install the prerequisite packages during installation:

```
# yum list <LIST_OF_PACKAGES>
```

For example:

```
# yum list httpd mod_ssl postgresql postgresql-devel postgresql-server
 compat-readline5 createrepo sigar sudo
```

If any of them are not available, then you may have not correctly added the repository to your admin host.

For the cluster hosts (where you plan to install the cluster), the prerequisite packages depend on the software you will eventually install there, but you may want to verify that the following two packages are installed or accessible by yum on all hosts:

- nc
- postgresql-devel

For the cluster hosts, the following packages need to be accessible if you are deploying in secure mode (the default):

- krb5-libs
- krb5-workstation
- openldap
- openldap-clients
- pam_krb5
- sssd
- authconfig
- openssh-clients
- python-ldap

Next Task:

PHD Pre-Inst 4 - Turn Off iptables on page 24

PHD Pre-Inst 4 - Turn Off iptables

Before you begin your installation, verify that iptables is turned off:

As `root`, run:

```
# chkconfig iptables off
# service iptables stop
```

Next Task:

PHD Pre-Inst 5 - Disable SELinux on page 24

PHD Pre-Inst 5 - Disable SELinux

Before you being your installation, verify that SELinux is disabled:

As `root`, run:

```
# sestatus
```

If SELinux is disabled, one of the following is returned:

```
SELinuxstatus: disabled
```

or

```
SELinux status: permissive
```

Disabling SELinux Temporarily

If SELinux status is **enabled**, you can temporarily disable it or make it permissive (this meets requirements for installation) by running the following command:

As `root`, run:

```
# echo 0 > /selinux/enforce
```

Note: This only temporarily disables SELinux; once the host is rebooted, SELinux will be re-enabled. We therefore recommend permanently disabling SELinux, described below, while running Pivotal HD/HAWQ (however, this requires a reboot).

Disabling SELinux Permanently

You can permanently disable SELinux by editing the `/etc/selinux/config` file as follows:

Change the value for the SELINUX parameter to:

`SELINUX=disabled`

Then reboot the system.

Next Task:

If you need to set up an *EPEL Yum Repository* on page 26, do so now.

Otherwise you have met all of the prerequisites, and can now proceed with *Installing PHD Using the CLI* on page 28.

sudo Configuration Files

The sudo configurations in `/etc/sudoers.d/gpadmin` are used for the `gpadmin` user to perform deployments and upgrades. This sudo configuration file is automatically created as part of the `preparehosts` command that is run during deployments and upgrades.

If you don't use the configuration files under `/etc/sudoers.d` due to your site security policy, you need to add the following sudo settings to your sudo configuration file to allow the `gpadmin` user to perform deployment and upgrade tasks. This needs to be done before attempting to deploy or upgrade.

```
####
Defaults:root,%gpadmin !requiretty


Defaults:root,%gpadmin secure_path += /sbin:/bin:/usr/sbin:/usr/bin:/usr/
local/bin


## Networking
Cmnd_Alias PCC_SYSTEM_NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /
sbin/dhclient, /sbin/iptables


### Installation and management of software
Cmnd_Alias PCC_SYSTEM_SOFTWARE = /bin/cp, /bin/mv, /bin/mkdir, /bin/grep, /
usr/bin/tee, /sbin/sysctl, /bin/chmod, /bin/chown, /bin/rpm, /usr/bin/
yum, /usr/bin/puppet, /usr/bin/createrepo, /usr/bin/ssh-keygen, /usr/sbin/
setenforce, /usr/sbin/useradd, /usr/sbin/ntpdate, /usr/bin/test, /usr/sbin/
alternatives, /usr/sbin/authconfig
```

```
### Commands with specific params
Cmnd_Alias PCC_COMMANDS_SPECIFIC_PARAMS = /bin/rm -rf /etc/gphd/*, /bin/rm
 -rf /etc/security/phd/*, /bin/rm -rf /usr/lib/gphd/*, /bin/rm -rf /var/lib/
gphd/*, /bin/rm -rf /var/log/gphd/*, /bin/rm -rf /tmp/.massh-gpadmin, /bin/
rm -rf ~gpadmin/*


### Services
Cmnd_Alias PCC_SYSTEM_SERVICES = /sbin/service, /sbin/chkconfig


### PCC specific services
Cmnd_Alias PCC_SERVICES = /etc/init.d/hadoop-hdfs-namenode, /etc/init.d/
hadoop-hdfs-datanode, /etc/init.d/hadoop-hdfs-secondarynamenode, /etc/
init.d/hadoop-yarn-resourcemanager, /etc/init.d/hadoop-yarn-nodemanager, /
etc/init.d/hadoop-mapreduce-historyserver, /etc/init.d/zookeeper-server, /
etc/init.d/hbase-master, /etc/init.d/hbase-regionserver, /etc/init.d/hive-
server, /etc/init.d/hive-metastore, /etc/init.d/postgresql, /etc/init.d/
hawq, /etc/init.d/uss-namenode, /home/gpadmin/jdk, /etc/init.d/hadoop-hdfs-
journalnode, /etc/init.d/hadoop-hdfs-zkfc, /etc/init.d/nodeagent, /etc/
init.d/zabbix-agent, /etc/init.d/pxf-service


### ICM Preparehost scripts
Cmnd_Alias PCC_PREPAREHOST_CMDS = /tmp/gphdgmr/addHawqConfigs.py


%gpadmin ALL=(root) NOPASSWD: PCC_SYSTEM_SOFTWARE, PCC_SYSTEM_SERVICES,
 PCC_SYSTEM_NETWORKING, PCC_SERVICES, PCC_COMMANDS_SPECIFIC_PARAMS,
 PCC_PREPAREHOST_CMDS


%gpadmin ALL=(hadoop,hdfs,mapred,yarn,hbase,hive,zookeeper,postgres)
 NOPASSWD:ALL



#####
```

Fully Qualified Domain Names (FQDN)

Make sure that your hostnames are fully qualified domain names (FQDN)

You can either:

1. Use the `hostname` command to set the FQDN:

   ```
   hostname www.example.com
   ```

   This is for live system updates only, and remains in effect only until the next reboot.

   or:
2. Change the value in `/etc/sysconfig/network` for changes to persist across reboots.

EPEL Yum Repository

Pivotal Command Center and Pivotal HD Enterprise expect some prerequisite packages to be pre-installed on each host, depending on the software that gets deployed on a particular host. In order to have a smoother installation, we recommend that each host have yum access to an EPEL yum repository. If you have access to the Internet, then you can configure your hosts to have access to the external EPEL repositories. However, if your hosts do not have Internet access (or you are deploying onto a large cluster) or behind a firewall, then having a local yum EPEL repository is highly recommended. This also gives you some control on the package versions you want to deploy on your cluster.

Following are the steps to create a local yum repository from a RHEL or CentOS DVD:

1. Mount the RHEL/CentOS DVD on a machine that will act as the local yum repository.

2. Install a webserver on that machine (e.g. httpd), making sure that HTTP traffic can reach this machine.

3. Install the following packages on the machine:

```
yum-utils
createrepo
```

4. Go to the directory where the DVD is mounted and run the following command:

```
# createrepo ./
```

5. Create a repo file on each host with a descriptive filename in the /etc/yum.repos.d/ directory of each host (for example, CentOS-6.1.repo) with the following contents:

```
[CentOS-6.1]
name=CentOS 6.1 local repo for OS RPMS
baseurl=http://172.254.51.221/centos/$releasever/os/$basearch/
enabled=1
gpgcheck=1
gpgkey=http://172.254.51.221/centos/$releasever/os/$basearch/RPM-GPG-KEY-
CentOS-6
```

6. Validate that you can access the local yum repos by running the following command:

```
# yum list
```

You can repeat the above steps for other software. If your local repos don't have any particular rpm, download one from a trusted source on the internet, copy it to your local repo directory and rerun the `createrepo` step.

Chapter 4

Installing PHD Using the CLI

This section describes how to install and configure Pivotal HD using command line interface (CLI) of Pivotal Command Center (PCC).

PHD Installation Checklist

The table below briefly describes the tasks you must complete to install PHD.

Each task is explained in more detail in subsequent sections; click the task name to jump to those sections.

Step	Task	Details	Completed

1	*Install Pivotal Command Center*	As `root`:
		1. Create a directory (`phd`) for your PCC installation:
		`# mkdir phd`
		2. Copy the tar file to your specified directory on the admin node. For example:
		`# scp ./PCC-2.3.x.version.build.os.x86_64.tar.gz host:/root/phd/`
		3. Log in as `root` and untar to that directory:
		`# cd /root/phd`
		`# tar --no-same-owner -zxvf PCC-2.3.x.version.build.os.x86_64.tar.gz`
		4. Run the installation script from the directory where it was extracted:
		`# ./install`
		5. As the rest of the installation is done as the `gpadmin` user, change to that user:
		`# su - gpadmin`
		6. If necessary, enable Secure Connections.	

2	*Configure Kerberos and LDAP*	On the Admin node, as `gpadmin`, run: `$ icm_client security -i` You will be prompted through the steps to set up a Kerberos Server if you don't have one for secure cluster configuration.	

Step	Task	Details	Completed

3	*Import the Packages*	Download and copy the PHD and related packages to the Admin node, then import the packages, including a downloaded JDK package, to the Admin node.
		As `gpadmin`:
		Copy the Packages:
		1. Copy the tarballs for the Pivotal HD services (PHD, ADS for HAWQ, and PRTS for GemFire XD) from the initial download location to the `gpadmin` home directory (`home/user/gpadmin`).
		2. Change the owner of the packages to `gpadmin`, then untar the tarballs.
		For example, if the file is a `tar.gz` or `.tgz` file, use:
		`tar -zxf packagename.tgz`
		If the file is a `.tar` file, use:
		`tar -xf packagename.tar`
		Import the Packages:
		Deploy the downloaded JDK to the cluster nodes
		`$ icm_client import -r <PATH_TO_JDK>`
		For each service (PHD, ADS, PRTS) you are importing, run the following:
		`$ icm_client import -s <PATH_TO_EXTRACTED_SERVICE_TARBALL>`	

4	*Edit the Cluster Configuration Files*	As `gpadmin`: **1.** Fetch the default Cluster Configuration template: `$ icm_client fetch-template -o ~/ClusterConfigDir` **2.** Edit the default Cluster Configuration template. At a minimum, you must replace all instances of your selected services with valid hostnames for your deployment. **Note:** **Gemfire XD**: If you want to use GemFire XD, you need to add that service to the `clusterConfig.xml` file. Also, GemFire XD may fail if it is not co-located with Hive. **High Availability**: As of PHD 2.1, High Availability is enabled by default. If you want to disable HA, you need to make some HA-specific changes to the `clusterConfig.xml` file and additionally edit some other configuration files (this can be done during installation or after). Complete instructions are available in the *High Availability* on page 125section.	

Step	Task	Details	Completed
		Security: If you are enabling security, there are some security-specific changes you need to make to the configuration file. Details are provided in *PHD Install 4 - Edit the Cluster Configuration Files* on page 35 **3.** **(Optional)** Edit the Hadoop services configuration files. Configure the other stack components in their corresponding configuration files as needed.	
5	*Edit the HAWQ Configuration File*	HAWQ system configuration is defined in `hawq/gpinitsystem_config`. Edit this file as needed.	
6	*PXF with GemFire XD*	Add `'/usr/lib/gphd/gfxd/lib/gemfirexd.jar'` on a new line to `ClusterConfigDir/pxf/pxf-public.classpath`.	
7	*Deploy the Cluster*	As `gpadmin`, run: `$ icm_client deploy -c ~/ClusterConfigDir` **Note:** This command creates the `gpadmin` user on the cluster nodes. Do NOT create this user manually. If `gpadmin` already exists on the cluster nodes, delete the user before running this command.	
8	*Start the Cluster*	As `gpadmin`, run: `$ icm_client start -l <CLUSTERNAME>`	

9	*Initialize and Start HAWQ*	As `gpadmin`:
		1. First verify HDFS is running:
		`$ ssh <NAME_NODE>`
		`$ hdfs dfs -ls /`
		2. Then, `ssh` to the HAWQ master, exchange keys, and run:
		`$ source /usr/local/hawq/greenplum_path.sh`
		`$ gpssh-exkeys -f HAWQ_HOSTS.txt`
		`$ /etc/init.d/hawq init`
		Where `HAWQ_HOSTS.txt` contains a list of HAWQ nodes.
		3. If you have a HAWQ standby master configured, initialize it using `gpinitstandby`.
		`gpinitstandby` reads the master data directory location from the `$MASTER_DATA_DIRECTORY` environment variable, so first run:
		`$ export MASTER_DATA_DIRECTORY=<MASTER_DIRECTORY>/gpseg-1`	

Step	Task	Details	Completed
		Then, run: `$ gpinitstandby -s <STANDBY_HAWQ_MASTER_FQDN>`	

PHD Install 1 - Install Pivotal Command Center

Perform the following installation steps as the `root` user.

Note: Avoid using hostnames that contain capital letters because Puppet has an issue generating certificates for domains with capital letters.

Avoid using underscores, as they are invalid characters in hostnames.

1. Download the PCC package from *Pivotal Network*.
2. As `root` on the Admin node, create a directory (`phd`) for your PCC installation on the Admin node:

```
# mkdir phd
```

3. Copy the Pivotal Command Center tar file to the Admin node, for example:

```
# scp ./PCC-2.3.x.version.build.os.x86_64.tar.gz host:/root/phd/
```

4. As `root`, `cd` to the directory where the Command Center tar files are located and untar them. For example:

```
# cd /root/phd
# tar --no-same-owner -zxvf PCC-2.3.x.version.build.os.x86_64.tar.gz
```

5. Still as `root` user, run the installation script. This installs the required packages, configures Pivotal Command Center, and starts services.

Important: You must run the installation script from the directory where it was extracted; for example: For example: `PCC-2.3.x.version`

For example:

```
# cd PCC-2.3.x.version
# ./install
```

You will see installation progress information on the screen.

You are given the option via a prompt during installation to specify a custom home directory for `gpadmin`. Before you deploy a cluster make sure that this home directory is consistent across all cluster hosts.Once the installation successfully completes, you will receive an installation success message on your screen.

6. Enable Secure Connections (optional):Pivotal Command Center uses HTTPS to secure data transmission between the client browser and the server. By default, the PCC installation script generates a self-signed certificate. Alternatively, you can provide your own Certificate and Key by following these steps:

   a. Set the ownership of the certificate file and key file to `gpadmin`.
   b. Change the permission to owner read-only (mode 400).
   c. Edit the `/etc/httpd/conf.d/pcc- vhost.conf` file and change the following two directives to point to the location of the SSL certificate and key. For example:

```
SSLCertificateFile:
```

```
/usr/local/pivotal-cc/ssl/<servername>.cert
SSLCertificateKeyFile:
/usr/local/pivotal-cc/ssl/<servername>.key
```

d. Restart PCC by running:

```
# service commander restart
```

Note: See *SSL Certificates* for details.

7. Verify that your PCC instance is running:

```
# service commander status
```

The PCC installation you just completed includes a CLI (Command Line Interface tool: `icm_client`). You can now deploy and manage the cluster using this CLI tool.

You can switch to the `gpadmin` user (created during installation) for the rest of the installation process:

```
$ su - gpadmin
```

Note: If, during the installation of PCC, you receive a facter mismatch error such as the following:

```
PCC-2.3.0-175]# rpm -ev facter
error: Failed dependencies:
facter >= 1.5 is needed by (installed) puppet-2.7.9-1.el6.noarch
```

Remove facter using the command:

```
yum erase facter
```

Then run the PCC installation again.

Next Task:

PHD Install 2 - Configure Kerberos and LDAP on page 32

PHD Install 2 - Configure Kerberos and LDAP

[Optional]

Kerberos is a network authentication protocol that provides strong authentication for client/server applications using secret-key cryptography.

You can configure PHD clusters to use Kerberos authentication.

Initializing security includes setting up a Kerberos server.

Note: If you already have a Kerberos server set up, you do not need to run the following command to initiate security, but you need to make security-specific edits to the cluster configuration file. SeeEditing the Cluster Configuration Files for details.

To initialize security:

1. On the Admin node, as `gpadmin`, run:

```
$ icm_client security -i
```

The installer will configure an internal LDAP/Kerberos server that will be used for the cluster-wide user management feature.

2. You will be prompted to specify whether to configure the built-in Kerberos server:

   ```
   Do you wish to configure Kerberos Server? (y/n) [Yes]? yes
   ```

   Enter `no` if you do not wish to use the built-in Kerberos server. The remaining instructions assume you chose to configure the built-in Kerberos server.
3. Choose a realm for your Kerberos server; usually this will be your domain name. For example:

   ```
   Enter REALM for Kerberos (ex PIVOTAL.IO): PIVOTAL.IO
   ```
4. Choose a login and password for your Kerberos server. You will need these if you ever need to manage the Kerberos server directly via the command line tool (`kadmin`). We recommend using `gpadmin`:

   ```
   Enter username for Kerberos Server ADMIN [admin]: gpadmin
   Enter new password for Kerberos Server ADMIN:
   Re-enter the new password for Kerberos Server Admin:
   Enter new MASTER password for KDC:
   Re-enter new MASTER password for KDC:
   ```
5. You are now prompted to set up the built-in LDAP server:

   ```
   [WARNING] Attempt to re-configure previously configure LDAP server may
    result in data or functionality loss
   Do you wish to configure LDAP Server? (y/n) [Yes]? yes
   ```
6. Select a suitable base domain name (DN); usually this will be your domain name. For example:

   ```
   Enter Domain name for LDAP base DN (ex pivotal.io): pivotal.io
   ```
7. Choose a login and password for the LDAP administrator. You will need these to add new users into the system, and also it will be needed if you ever need to manage the built-in LDAP server directly. We recommend using `gpadmin`:

   ```
   Enter username for LDAP Administrator [Manager]: gpadmin
   Enter new password for LDAP administrator:
   Re-enter new password for LDAP administrator:
   ```
8. The installer will now install and configure the built-in Kerberos and LDAP server, based on the information you provided:

   ```
   [INFO] Attempting to configure KDC and/or LDAP. It may take few minutes...
   [DONE] Security components initialized successfully
   ```

Next Task:

PHD Install 3 - Import the PHD Service Packages on page 33

Note: If you chose to configure security, you need to made security-specific changes to the cluster configuration file. For more information, see *PHD Install 4 - Edit the Cluster Configuration Files* on page 35.

In addition, if you are also planning to install HAWQ, you need to make some post-installation changes to HAWQ. See *PHD Install 9 - Initialize and Start HAWQ* on page 40 for details.

PHD Install 3 - Import the PHD Service Packages

Once you have Pivotal Command Center installed, you can use the `import` option of the `icm_client` tool to synchronize the PHD service RPMs and a downloaded JDK package from the specified source location into the Pivotal Command Center (PCC) local yum repository of the Admin node. This allows the cluster nodes to access the packages during deployment.

If you need to troubleshoot this part of the installation process, see the log file located at: `/var/log/gphd/gphdmgr/gphdmgr-import.log`

Import JDK

Note that having JDK 1.7 running on the Admin node is a prerequisite. This step describes how to import a downloaded JDK package that will be deployed across the cluster:

1. Download a supported JDK package from *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. PHD expects an RPM package; for example: `jdk-7u45-linux-x64.rpm`
2. Import the downloaded JDK package to the cluster nodes. As `gpadmin`, run:

```
$ icm_client import -r <PATH_TO_JDK>
```

Copy the PHD Service Packages

1. Download the PHD service packages (PHD, and optionally ADS for HAWQ and PRTS for GemFire XD) from the *Pivotal Network*.
2. Copy the Pivotal HD (and ADS and PRTS if downloaded) tarballs from your initial download location to the `gpadmin` home directory on the Admin node (`home/gpadmin`).
3. Change the owner of the packages to `gpadmin` and untar the tarballs. For example:

```
# For PHD: If the file is a tar.gz or tgz, use
$ tar zxf PHD-2.1.x-<BUILD>.tar.gz

# If the file is a tar, use
$ tar xf PHD-2.1.x-<BUILD>.tar

# For Pivotal ADS: If the file is a tar.gz or tgz file, use
$ tar zxf PADS-1.2.x-<BUILD>.tar.gz

# If the file is a tar, use
$ tar xf PADS-1.2.x-<BUILD>.tar

# For PRTS: If the file is a tar.gz or tgz file, use
$ tar zxf PRTS-1.x.x-<BUILD>.tar.gz

# If the file is a tar, use
$ tar xf PRTS-1.x.x-<BUILD>.tar
```

Import PHD Service

1. As `gpadmin`, import the following tarball for Pivotal HD:

```
$ icm_client import -s <PATH_OF_EXTRACTED_PHD_PACKAGE>
```

For example:

```
$ icm_client import -s PHD-2.0.x-x/
```

Import HAWQ/PXF Services

[Optional]

As `gpadmin`, import the following tarballs for HAWQ and PXF:

```
$ icm_client import -s <PATH_OF_EXTRACTED_ADS_PACKAGE>
```

For example:

```
$ icm_client import -s PADS-1.2.x-x/
```

Import PRTS (GemFire XD) Service

[Optional]

As `gpadmin`, import the following tarball for PRTS:

```
$ icm_client import -s <PATH_OF_EXTRACTED_PRTS_PACKAGE>
```

For example:

```
$ icm_client import -s PRTS-1.x.x-x/
```

Next Task:

PHD Install 4 - Edit the Cluster Configuration Files on page 35

PHD Install 4 - Edit the Cluster Configuration Files

Pivotal provides a default Cluster configuration file (`clusterConfig.xml`) that you need to edit for your own cluster; all the cluster nodes are configured based on this configuration file.

At a minimum, you must replace all instances of your selected services with valid hostnames for your deployment.

Advanced users can further customize their cluster configuration by editing the stack component configuration files such as `hdfs/core-site.xml`.

Important:

- Always use fully-qualified domain names (FQDN), rather than short hostnames, in the `clusterConfig.xml` file.
- For more information about setting a FQDN, see *Fully Qualified Domain Names (FQDN)* on page 26.

Fetch the Default Cluster Configuration Template

The `fetch-template` command saves a default cluster configuration template into a specified directory, such as a directory on disk. You can then manually modify this template and use it as input to subsequent commands.

As `gpadmin`, run the `fetch-template` command. For example:

```
$ icm_client fetch-template -o ~/ClusterConfigDir
```

This example uses the `fetch-template` command to place a template in a directory called `ClusterConfigDir` (automatically created by the command). This directory contains files that describe the topology of the cluster and the configurations for the various services installed on the cluster.

Configure HBase Bulk Loading in Secure Mode

1. Assuming your clusterconfig template directory on the admin node is `ClusterConfigDir`, change the `hbase-auth` in `ClusterConfigDir/security/security-driver.xml` to `true`.

2. Optionally, you may configure `hadoop` group or a list of users such as `gpadmin` as `hbase.superuser`. To do this, add the following to your `ClusterConfigDir/hbase/hbase-site.xml` file:

```
<!-- this example adds all users under hadoop group and user 'myorgadmin'
 as superuser for hbase
<property>
     <name>hbase.superuser</name>
     <value>@hadoop, myorgadmin</value>
</property>
```

3. Continue configuring your cluster, as described below.
4. If you did not perform step 2 above, once you have deployed your cluster, on the Hbase master, grant user `gpadmin` or any other user who you want to bulk load with permissions to create tables. For more details see the *PHD Stack and Tool Reference*.

   For more details about bulk loading, see, *http://hbase.apache.org/book/arch.bulk.load.html*.

Edit the clusterConfig.xml file

Edit the `clusterConfig.xml` file as follows:

1. Locate and edit the `clusterConfig.xml` file based on your cluster requirements. The following sections should be verified or edited:

   a. **Header section:** This is the metadata section and must contain the following mandatory information:

      - `clusterName`: The name of your cluster
      - `gphdStackVer`: Pivotal HD Version. Accepted values are: `PHD-2.0.1.0`, `PHD-2.0.0.0`, `PHD-1.1.1.0`, `PHD-1.1.0.0`
      - `services`: Configure the services to be deployed. By default, every service that Pivotal HD supports is listed here. ZooKeeper, HDFS, and YARN are mandatory services. HBase and HAWQ are optional.
      - `client`: The host that can be used as a gateway or launcher node for running the Hadoop, Hive, Pig, and Mahout jobs.

   b. **Topology Section** `<HostRoleMapping>`**:** This is the section where you specify the roles to be installed on the hosts. For example, you can specify where your Hadoop NameNode, DataNode, etc. should be installed. Note that all mandatory roles should have at least one host allocated. You can identify the mandatory role by looking at the comment above that role in the `clusterConfig.xml` file.

   c. **Global Service Properties** `<servicesConfigGlobals>`**:** This section defines mandatory global parameters such as Mount Points, Directories, Ports, and `JAVA_HOME`. These configured mount points such as `datanode.disk.mount.points`, `namenode.disk.mount.points`, and `secondary.namenode.disk.mount.points` are used to derive paths for other properties in the DataNode, NameNode and SecondaryNameNode configurations, respectively. These properties can be found in the individual service configuration files.

      For Secure Clusters: If you want to deploy secure clusters, you must have first initialized security (see *PHD Install 2 - Configure Kerberos and LDAP* on page 32), then make the following changes to the **Global Services Properties** section:

      i. Locate the following sub-section within the **Global Services Properties** section:

```
<!-- Security configurations -->
 <!-- provide security realm. e.g. EXAMPLE.COM -->
 <security.realm></security.realm>
 <!-- provide the path of kdc conf file -->
 <security.kdc.conf.location>/etc/krb5.conf</
security.kdc.conf.location>
```

ii. You need to add a valid value to the `<security.realm>` parameter.The default value for the `<security.kdc.conf.location>` parameter is valid if you used the Kerberos server that was set up during *PHD Install 2 - Configure Kerberos and LDAP* on page 32; if you are using an existing Kerberos server, you need to add a value for that location.

Important:

The following information pertains to configuring the parameters in the **Global Services Properties** section:

- `hawq.segment.directory` and `hawq.master.directory` need to be configured only if HAWQ is used.
- The values in this section are pre-filled with defaults. Check these values; they may not need to be changed.
- The directories specified in the mount points will be automatically created by PCC while deploying PHD, if they don't already exist.
- Pivotal recommends that you have multiple disk mount points for datanodes, but it is not a requirement.

d. GemFire XD:

- If you want to use GemFire XD, you need to add that service to the `clusterConfig.xml` file.
  - Add `gfxd` to the services listed in the `<services></services>` tag.
  - Define the `gfxd-server` and `gfxd-locator` roles in the `clusterConfig.xml` file for every cluster by adding the following to the `<hostrolemapping> </hostrolemapping>` tag:

```
<gfxd>
   <gfxd-locator>host.yourdomain.com</gfxd-locator>
   <gfxd-server>host.yourdomain.com</gfxd-server>
</gfxd>
```

- If you have PXF using GFXD as a data source, add `'/usr/lib/gpdb/gfxd/lib/gemfirexd.jar'` on a new line to `ClusterConfigDir/pxf/pxf-public.classpath`.

You cannot start GemFire XD (`gfxd`) using the `icm_client start` command. Refer to the GemFire XD documentation (*http://gemfirexd.docs.pivotal.io/latest/userguide/index.html?q=getting_started/book_intro.html*) for instructions about starting your `gfxd` service.

Note: HAWQ and GFXD services are both memory intensive and it is best to configure these services to be deployed on different nodes.

GemFire XD may fail if it is not co-located with Hive.

e. High Availability: As of PHD 2.1, high availability is enabled by default. If you want to disable HA, you need to make some HA-specific changes to the `clusterConfig.xml` file and additionally edit some other configuration files (this can be done during installation or after). Complete instructions are available in the *High Availability* on page 125 section.

f. Security: Security is enabled by default and is specified by the followign configuration file parameter/value:

```
<securityEnabled>true</securityEnabled>
```

For more information about securing clusters, see *PHD Install 2 - Configure Kerberos and LDAP* on page 32.To disable security, change this value to `false`.

2. Once you've made your changes, we recommend you check that your xml is well-formed using the `xmlwf` command, as follows:

   ```
   xmlwf ~/ClusterConfigDir/clusterConfig.xml
   ```

3. Save and close the `clusterConfig.xml` file.

Edit the Hadoop Services Configuration Files

Most Hadoop services have a corresponding directory that contains their standard configuration file(s). You can edit/change properties to suit your cluster requirements, or consult with Pivotal HD support to decide on a configuration to suit your specific cluster needs.

Note: If the directories specified in `dfs.namenode.name.dir` and `dfs.datanode.data.dir` in the `hdfs/hdfs-site.xml` pre-exist, then they should be empty.

Note: You must not override properties derived from the global service properties, especially those derived from role/hostname information.

Next Task:

PHD Install 5 - Edit the HAWQ Configuration File on page 38

PHD Install 5 - Edit the HAWQ Configuration File

HAWQ system configuration is defined in `hawq/gpinitsystem_config`.

- You can override the HAWQ database default database port setting, 5432, using the `MASTER_PORT` parameter.
- You can also change the HAWQ DFS path using the `DFS_URL` parameter.

Important: Memory/VMs Issue

If you are planning to deploy a HAWQ cluster on VMs with memory lower than the optimized/ recommended requirements, do the following:

1. Prior to deploying your cluster, open the `/usr/lib/gphd/gphdmgr/hawq_sys_config/sysctl.conf` file and change the value of the following parameter from 2 to 0: `vm.overcommit_memory =0`.
2. In `clusterConfig.xml`, update `<hawq.segment.directory>` to include only one segment directory entry (instead of the default 2 segments).

Next Task:

- If needed: *PHD Install 6 - PXF with GemFire XD* on page 38
- Otherwise, skip to: *PHD Install 7 - Deploy the Cluster* on page 39

PHD Install 6 - PXF with GemFire XD

If you have PXF using GemFire XD (GFXD) as a data source, add `'/usr/lib/gphd/gfxd/lib/gemfirexd.jar'` on a new line to `ClusterConfigDir/pxf/pxf-public.classpath`.

Next Task:

PHD Install 7 - Deploy the Cluster on page 39

PHD Install 7 - Deploy the Cluster

Pivotal HD deploys clusters using input from the cluster configuration directory. This cluster configuration directory contains files that describes the topology and configuration for the cluster.

Deploy the cluster as `gpadmin`.

The `deploy` command internally performs three tasks:

1. Prepares the cluster nodes with the prerequisites (internally runs `preparehosts` command)
   a. Creates the `gpadmin` user.
   b. As `gpadmin`, sets up password-less SSH access from the Admin node.
   c. Installs the provided Oracle Java JDK.
   d. Disables SELinux across the cluster.
   e. Optionally synchronizes the system clocks.
   f. Installs Puppet version 2.7.20 (the one shipped with the PCC tarball, not the one from puppetlabs repo).
   g. Installs sshpass.
   h. Disables iptables across the cluster.
2. Verifies the prerequisites (internally runs `scanhosts` command).
3. Deploys the cluster.

Note: `scanhosts` and `preparehosts` are commands that in previous releases you could run independently. Starting with release 2.0.1 they are run internally as part of the `deploy` command. As such, these commands are deprecated and should not be run independently.

Note: Deploying multiple clusters at the same time is not supported; deploy one cluster at a time.

For example:

```
$ icm_client deploy -c -t ClusterConfigDir/ -i -d -j jdk-7u15-linux-
x86_64.rpm
```

You can check the following log files to troubleshoot any failures:

On Admin:

`/var/log/gphd/gphdmgr/GPHDClusterInstaller_XXX.log`

`/var/log/gphd/gphdmgr/gphdmgr-webservices.log`

`/var/log/messages`

`/var/log/gphd/gphdmgr/installer.log`

`/var/log/gphd/tools/security/icm_integration.log`

On Cluster Nodes:

`/tmp/GPHDNodeInstaller_XXX.log`

icm_client deploy Syntax:

```
icm_client deploy --help
Usage: /usr/bin/icm_client deploy [options]

Options:
  -h, --help            show this help message and exit
  -c CONFDIR, --confdir=CONFDIR
```

```
                       Directory path where cluster configuration is stored
 -s, --noscanhosts     Do not verify cluster nodes as part of deploying the
                       cluster
 -p, --nopreparehosts  Do not prepare hosts as part of deploying the
cluster
 -j JDKPATH, --java=JDKPATH
                       Location of Sun Java JDK RPM (Ex: jdk-
                       7u15-linux-x64.rpm). Ignored if -p is specified
 -t, --ntp             Synchronize system clocks using NTP. Optionally
takes
                       NTP server as argument. Defaults to pool.ntp.org
                       (requires external network access). Ignored if -p is
                       specified
 -d, --selinuxoff      Disable SELinux for the newly added nodes. Ignored
if -p is specified
 -i, --iptablesoff     Disable iptables for the newly added nodes. Ignored
if -p is specified
 -P, --nopasswordlessssh
                       Skip setting up passwordless ssh for gpadmin
account.
                       This assumes the passwordless ssh has already been
                       setup beforeahead. Ignored if -p is specified.
 -y SYSCONFIGDIR, --sysconf=SYSCONFIGDIR
                       [Only if HAWQ is part of the deploy] Directory
                       location of the custom conf files (sysctl.conf and
                       limits.conf) which will be appended to
                       /etc/sysctl.conf and /etc/limits.conf on slave
nodes.
                       Default: /usr/lib/gphd/gphdmgr/hawq_sys_config/.
                       Ignored if -p is specified
```

Your Pivotal HD installation is complete.

Next Task:

PHD Install 8 - Start the Cluster on page 40

PHD Install 9 - Initialize and Start HAWQ on page 40

PHD Install 8 - Start the Cluster

As `gpadmin`, use `icm_client` to start your cluster.

For example:

```
$ icm_client start -l <CLUSTERNAME>
```

See *Starting a Cluster* on page 112 for more detailed instructions and other startup options.

Next Task:

If you are using HAWQ: *PHD Install 9 - Initialize and Start HAWQ* on page 40

PHD Install 9 - Initialize and Start HAWQ

Initializing HAWQ performs the following tasks:

- Initializes the HAWQ master and the segment hosts.
- Starts the HAWQ master, segments, and the underlying postgres database.

You need to initialize HAWQ only once, after the cluster has started and after HDFS is up and running.

Note:

Verify that the `postgres` user exists. If it does not, you may have to create it and add it into the `hadoop` group.

To initialize HAWQ:

1. **Security:** If you have deployed a secure cluster with Kerberos authentication, you must create a Kerberos principal for `gpadmin` and run `kinit` before running the next command.

   Note: If you have not deployed a secure cluster, skip this task.

   To add a principal for `gpadmin`:

   a. On the PCC Admin node, run:

   ```
   $ sudo kadmin.local
   $ add princ gpadmin
   $ exit
   ```

   Note: Provide a password for the `gpadmin` principal when prompted.

   b. Run:

   ```
   $ kinit
   ```

2. Verify HDFS is running.

   To verify HDFS is running, log in to the client node, NameNode or DataNode as `gpadmin` and run:

   ```
   $ hdfs dfs -ls /
   ```

   Sample Output:

   ```
     Found 4 items
     drwxr-xr-x   - mapred hadoop          0 2013-06-15 15:49 /mapred
     drwxrwxrwx   - hdfs   hadoop          0 2013-06-15 15:49 /tmp
     drwxrwxrwx   - hdfs   hadoop          0 2013-06-15 15:50 /user
     drwxr-xr-x   - hdfs   hadoop          0 2013-06-15 15:50 /yarn
   ```

3. **Security:** If you have deployed a secure cluster with Kerberos authentication:

   Note: If you have not deployed a secure cluster, skip this task.

   a. Locate the HAWQ data directory:

      i. On the HAWQ master, open `/etc/gphd/hawq/conf/gpinitsystem_config`.

      ii. Locate `DFS_URL` and obtain the directory after `nameservice` or `namenode`. By default the value of this is `hawq_data`. We will refer to it as <*HAWQ_DATA_DIR*> for the purpose of this document.

   b. Create <*HAWQ_DATA_DIR*> on HDFS:

      i. Start the cluster using `icm_client`.

      ii. Make sure HDFS service is up and running.

      iii. As `gpadmin`, on the namenode or client machine, run:

      ```
      kinit
      hadoop fs -mkdir /<HAWQ_DATA_DIR>
      hadoop fs -chown -R postgres:gpadmin /<HAWQ_DATA_DIR>
      hadoop fs -mkdir /user/gpadmin
      ```

```
hadoop fs -chown gpadmin:gpadmin /user/gpadmin
hadoop fs -chmod 777 /user/gpadmin
kdestroy
```

4. As `gpadmin`, exchange keys, then initialize HAWQ from the HAWQ master.

   Note: `ssh` to the HAWQ Master before you initialize HAWQ.

   For example:

   ```
   $ su - gpadmin
   $ source /usr/local/hawq/greenplum_path.sh
   $ gpssh-exkeys -f HAWQ_HOSTS.txt # where HAWQ_HOSTS.txt has a set of hawq
    nodes
   $ /etc/init.d/hawq init
   ```

   Note: You do not need to start HAWQ. It is implicitly started as part of the initialization.

5. If you have a HAWQ Standby master in your cluster configuration, initialize it using `gpinitstandby`:

   a. `gpinitstandby` reads the master data directory location from the `$MASTER_DATA_DIRECTORY` environment variable, so before running `gpinitstandby`, run the following:

   ```
   $ export MASTER_DATA_DIRECTORY=<MASTER_DIRECTORY>/gpseg-1
   ```

   For example:

   ```
   $ export MASTER_DATA_DIRECTORY=/data0/master/gpseg-1/gpseg-1
   ```

   b. Then, still as `gpadmin`, initialize the standby master:

   ```
   $ gpinitstandby -s <HAWQ_STANDBY_MASTER_FQDN>
   ```

Note: Hive with HAWQ/PXF

If you are planning to configure Hive with HAWQ/PXF, check that the Hive Metastore service is available and running (anywhere on the cluster) and that you have set the property `hive.metastore.uri` in the `hive-site.xml` file on the NameNode to point to that location.

For more information about HAWQ administration, see *Managing HAWQ*.

Next Task:

None. Your PHD/HAWQ installation is now complete.

However, there are some post-installation tasks you should consider next, such as verifying services and running sample programs. For more information, see *PHD Post-Install* on page 44.

Chapter 5

PHD Post-Install

This section describes tasks you perform after installing PHD.

Verifying PHD Service Status

You can use the `service status` command to check the running status of a particular service role from its appropriate host(s).

Refer to *Running PHD Sample Programs* on page 44 where you can see the sample commands for each Pivotal HD service role.

The following example shows an aggregate status view of Hadoop, Zookeeper and HBase service roles from all the cluster nodes:

```
[gpadmin]\# massh ./HostFile.txt verbose 'sudo service --status-all | egrep
 "hadoop | zookeeper | hbase"
```

Below is an example to check the status of all datanodes in the cluster:

```
# Create a newline separated file named 'datanodes.txt' containing all the
 datanode belonging to the service role \\
[gpadmin]\# massh datanodes.txt verbose 'sudo service hadoop-hdfs-datanode
 status'
```

Running PHD Sample Programs

Make sure you are logged in as user `gpadmin` on the appropriate host before testing any of the services.

- *Testing Hadoop*
- *Testing YARN*
- *Testing Zookeeper*
- *Testing HBase and ZooKeeper*
- *Testing HAWQ*
- *Testing Pig*
- *Testing Hive*
- *Testing Hcatalog*
    - *Using HCatalog Command-line API*
    - *Using HCatalog with REST*
    - *Using HCatalog with Pig*
- *Testing Oozie*
    - *Submit Oozie Example Workflows*
- *Testing Sqoop*

Testing Hadoop

You can run Hadoop commands from any configured Hadoop nodes.You can run MapReduce jobs from the DataNodes, resource manager, or historyserver.

```
# clear input directory, if any |

$ hadoop fs -rmr /tmp/test_input

# create input directory
$ hadoop fs -mkdir /tmp/test_input

# ensure output directory does not exist
$ hadoop fs -rmr /tmp/test_output

# copy some file having text data to run word count on
$ hadoop fs -copyFromLocal /usr/lib/gphd/hadoop/CHANGES.txt /tmp/test_input

# run word count
$ hadoop jar /usr/lib/gphd/hadoop-mapreduce/hadoop-mapreduce-examples-
<version>.jar wordcount /tmp/test_input /tmp/test_output

# dump output on console
$ hadoop fs -cat /tmp/test_output/part*
```

Note: When you run a MapReduce job as a custom user (i.e. not as `gpadmin`, `hdfs`, `mapred`, or `hbase`), note the following:

- Make sure the appropriate user staging directory exists.
- Set permissions on `yarn.nodemanager.remote-app-log-dir` to 777. For example, if it is set to the default value `/yarn/apps`, do the following:

  ```
  $ sudo -u hdfs hadoop fs -chmod 777 /yarn/apps
  ```

- Ignore the Exception trace, this is a known Apache Hadoop issue.

Testing YARN

```
Run a yarn job (Pi job):

yarn jar /usr/lib/gphd/hadoop-mapreduce/hadoop-mapreduce-examples-2.2.0-
gphd-3.1.0.0.jar pi 2 2

List all jobs with their status:

sudo -u hadoop yarn application -list
```

```
14/07/25 11:05:24 INFO client.RMProxy: Connecting to ResourceManager at
 centos64-2.localdomain/192.168.2.202:8032
Total number of applications (application-types: [] and states: [SUBMITTED,
 ACCEPTED, RUNNING]):1
                Application-Id      Application-Name        Application-Type
          User           Queue                   State             Final-
State         Progress                         Tracking-URL
application_1406286051207_0001       QuasiMonteCarlo               MAPREDUCE
       gpadmin         default                 RUNNING
 UNDEFINED               5%               http://centos64-2:7017


[gpadmin@centos64-2 ~]$ sudo -u hadoop yarn application -status
 application_1406286051207_0001
14/07/25 11:05:36 INFO client.RMProxy: Connecting to ResourceManager at
 centos64-2.localdomain/192.168.2.202:8032
Application Report :
        Application-Id : application_1406286051207_0001
        Application-Name : QuasiMonteCarlo
        Application-Type : MAPREDUCE
        User : gpadmin
        Queue : default
        Start-Time : 1406286289246
        Finish-Time : 0
        Progress : 5%
        State : RUNNING
        Final-State : UNDEFINED
        Tracking-URL : http://centos64-2:7017
        RPC Port : 21905
        AM Host : centos64-2
        Diagnostics :
[gpadmin@centos64-2 ~]$
```

Testing Zookeeper

To test Zookeeper, first make sure that Zookeeper is ruuning. Zookeeper responds to a small set of commands. Each command is composed of four letters. You issue commands to Zookeeper via telnet or nc, at the client port.

From any client nodes, use the following commands to check zookeeper :

```
ZooKeeper Commands: The Four Letter Words

[gpadmin@centos64-3 ~]$ echo ruok | nc localhost 2181
imok[gpadmin@centos64-3 ~]$
[gpadmin@centos64-3 ~]$ echo dump | nc localhost 2181
SessionTracker dump:
org.apache.zookeeper.server.quorum.LearnerSessionTracker@4ed78fd5
ephemeral nodes dump:
Sessions with Ephemerals (3):
0x1478ff8e66e0001:
        /hadoop-ha/test/ActiveStandbyElectorLock
0x1478ff8e66e0002:
        /hbase/master
        /hbase/tokenauth/keymaster
0x2478ff8e67c0001:
        /hbase/rs/centos64-3.localdomain,60020,1406869842986
[gpadmin@centos64-3 ~]$
[gpadmin@centos64-3 ~]$
[gpadmin@centos64-3 ~]$ echo envi | nc localhost 2181
Environment:
zookeeper.version=3.4.5--1, built on 07/03/2014 06:24 GMT
```

```
host.name=centos64-3.localdomain
java.version=1.7.0_45
java.vendor=Oracle Corporation
java.home=/usr/java/jdk1.7.0_45/jre
java.class.path=/usr/lib/gphd/zookeeper/bin/../build/classes:/usr/lib/
gphd/zookeeper/bin/../build/lib/*.jar:/usr/lib/gphd/zookeeper/bin/../
lib/slf4j-log4j12-1.6.1.jar:/usr/lib/gphd/zookeeper/bin/../lib/slf4j-
api-1.6.1.jar:/usr/lib/gphd/zookeeper/bin/../lib/netty-3.2.2.Final.jar:/
usr/lib/gphd/zookeeper/bin/../lib/log4j-1.2.16.jar:/usr/lib/gphd/zookeeper/
bin/../lib/jline-0.9.94.jar:/usr/lib/gphd/zookeeper/bin/../zookeeper-3.4.5-
gphd-3.1.0.0.jar:/usr/lib/gphd/zookeeper/bin/../src/java/lib/*.jar:/etc/
gphd/zookeeper/conf::/etc/gphd/zookeeper/conf:/usr/lib/gphd/zookeeper/
zookeeper-3.4.5-gphd-3.1.0.0.jar:/usr/lib/gphd/zookeeper/zookeeper.jar:/
usr/lib/gphd/zookeeper/lib/log4j-1.2.16.jar:/usr/lib/gphd/zookeeper/lib/
netty-3.2.2.Final.jar:/usr/lib/gphd/zookeeper/lib/slf4j-log4j12-1.6.1.jar:/
usr/lib/gphd/zookeeper/lib/jline-0.9.94.jar:/usr/lib/gphd/zookeeper/lib/
slf4j-api-1.6.1.jar
java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/
lib
java.io.tmpdir=/tmp
java.compiler=<NA>
os.name=Linux
os.arch=amd64
os.version=2.6.32-358.el6.x86_64
user.name=zookeeper
user.home=/home/zookeeper
user.dir=/home/gpadmin
[gpadmin@centos64-3 ~]$ echo stat | nc localhost 2181
Zookeeper version: 3.4.5--1, built on 07/03/2014 06:24 GMT
Clients:
 /192.168.2.203:5044[1](queued=0,recved=2842,sent=2842)
 /192.168.2.202:4723[1](queued=0,recved=1186,sent=1212)
 /0:0:0:0:0:0:0:1:18798[0](queued=0,recved=1,sent=0)
 /192.168.2.203:5058[1](queued=0,recved=364,sent=364)

Latency min/avg/max: 0/1/1676
Received: 4672
Sent: 4697
Connections: 4
Outstanding: 0
Zxid: 0x1800000040
Mode: follower
Node count: 53
[gpadmin@centos64-3 ~]$
[gpadmin@centos64-3 ~]$
```

Testing HBase and ZooKeeper

You can test HBase from the HBase master node.

To test zookeeper, from the HBase shell, run the `zk_dump` command:

```
gpadmin# ./bin/hbase shell
hbase(main):003:0> create 'test', 'cf'
0 row(s) in 1.2200 seconds
hbase(main):003:0> list 'test'
..
1 row(s) in 0.0550 seconds
hbase(main):004:0> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.0560 seconds
hbase(main):005:0> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0370 seconds
```

```
hbase(main):006:0> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0450 seconds

hbase(main):007:0> scan 'test'
ROW COLUMN+CELL
row1 column=cf:a, timestamp=1288380727188, value=value1
row2 column=cf:b, timestamp=1288380738440, value=value2
row3 column=cf:c, timestamp=1288380747365, value=value3
3 row(s) in 0.0590 seconds

hbase(main):012:0> disable 'test'
0 row(s) in 1.0930 seconds
hbase(main):013:0> drop 'test'
0 row(s) in 0.0770 seconds

hbase(main):002:0> zk_dump
HBase is rooted at /hbase
Active master address: centos64-2.localdomain,60000,1406799746730
Backup master addresses:
Region server holding hbase:meta: centos64-3.localdomain,60020,1406799753532
Region servers:
 centos64-2.localdomain,60020,1406799754233
 centos64-3.localdomain,60020,1406799753532
 centos64-4.localdomain,60020,1406799751248
/hbase/replication:
/hbase/replication/peers:
/hbase/replication/rs:
/hbase/replication/rs/centos64-4.localdomain,60020,1406799751248:
/hbase/replication/rs/centos64-3.localdomain,60020,1406799753532:
/hbase/replication/rs/centos64-2.localdomain,60020,1406799754233:
Quorum Server Statistics:
 centos64-3.localdomain:2181
  Zookeeper version: 3.4.5--1, built on 04/14/2014 03:32 GMT
  Clients:
   /192.168.2.202:24969[1](queued=0,recved=153,sent=153)
   /192.168.2.203:61845[1](queued=0,recved=150,sent=150)
   /192.168.2.202:24955[1](queued=0,recved=457,sent=488)
   /192.168.2.204:40463[1](queued=0,recved=150,sent=150)
   /192.168.2.204:40460[1](queued=0,recved=174,sent=177)
   /192.168.2.202:24968[1](queued=0,recved=181,sent=181)
   /192.168.2.202:25189[0](queued=0,recved=1,sent=0)

  Latency min/avg/max: 0/3/2432
  Received: 1266
  Sent: 1299
  Connections: 7
  Outstanding: 0
  Zxid: 0x10000006f
  Mode: follower
  Node count: 38
 centos64-2.localdomain:2181
  Zookeeper version: 3.4.5--1, built on 04/14/2014 03:32 GMT
  Clients:
   /192.168.2.202:21459[1](queued=0,recved=16,sent=16)
   /192.168.2.202:21458[1](queued=0,recved=5,sent=5)
   /192.168.2.203:13881[1](queued=0,recved=151,sent=151)
   /192.168.2.202:21462[0](queued=0,recved=1,sent=0)

  Latency min/avg/max: 0/6/720
  Received: 226
  Sent: 225
  Connections: 4
  Outstanding: 0
  Zxid: 0x10000006f
  Mode: follower
  Node count: 38
 centos64-4.localdomain:2181
  Zookeeper version: 3.4.5--1, built on 04/14/2014 03:32 GMT
  Clients:
   /192.168.2.203:40472[1](queued=0,recved=196,sent=198)
   /192.168.2.202:19701[1](queued=0,recved=189,sent=191)
   /192.168.2.202:19931[0](queued=0,recved=1,sent=0)
   /192.168.2.202:19712[1](queued=0,recved=150,sent=150)
   /192.168.2.202:19710[1](queued=0,recved=151,sent=151)
   /192.168.2.204:47427[1](queued=0,recved=150,sent=150)

  Latency min/avg/max: 0/0/27
  Received: 872
  Sent: 875
  Connections: 6
  Outstanding: 0
  Zxid: 0x10000006f
  Mode: leader
  Node count: 38

hbase(main):003:0>
```

Testing HAWQ

Note: Use the HAWQ Master node to run HAWQ tests.

```
gpadmin# source /usr/local/hawq/greenplum_path.sh

gpadmin# psql -p 5432
psql (8.2.15)
Type "help" for help.

gpadmin=# \d
No relations found.
gpadmin=# \l
List of databases
Name | Owner | Encoding | Access privileges
---{}----+------------
gpadmin | gpadmin | UTF8 |
postgres | gpadmin | UTF8 |
template0 | gpadmin | UTF8 |
template1 | gpadmin | UTF8 |
(4 rows)

gpadmin=# \c gpadmin
You are now connected to database "gpadmin" as user "gpadmin".
gpadmin=# create table test (a int, b text);
NOTICE: Table doesn't have 'DISTRIBUTED BY' clause -
Using column named 'a' as the Greenplum Database data
distribution key for this table.
HINT: The 'DISTRIBUTED BY' clause determines the distribution
of data. Make sure column(s) chosen are the optimal data
distribution key to minimize skew.

CREATE TABLE

gpadmin=# insert into test values (1, '435252345');
INSERT 0 1
gpadmin=# select * from test;
a | b
```

```
-+---------
1 | 435252345
(1 row)

gpadmin=#
```

Testing Pig

You can test Pig from a client node:

```
# Clean up input/output directories

hadoop fs -rmr /tmp/test_pig_input
hadoop fs -rmr /tmp/test_pig_output

#Create input directory

hadoop fs -mkdir /tmp/test_pig_input

# Copy data from /etc/passwd

hadoop fs -copyFromLocal /etc/passwd /tmp/test_pig_input
```

In the grunt shell, run this simple Pig job:

```
$ pig // Enter grunt shell
A = LOAD '/tmp/test_pig_input' using PigStorage(':');
B = FILTER A by $2 > 0;
C = GROUP B ALL;
D = FOREACH C GENERATE group, COUNT(B);
STORE D into '/tmp/test_pig_output';

# Displaying output

hadoop fs -cat /tmp/test_pig_output/part*

Cleaning up input and output'

hadoop fs -rmr /tmp/test_pig_*
```

Testing Hive

Test Hive from a client node:

```
gpadmin# hive

# Creating passwords table
hive> create table passwords (col0 string, col1 string, col2 string, col3
 string, col4 string, col5 string, col6 string) ROW FORMAT DELIMITED FIELDS
 TERMINATED BY ":";
hive> SHOW TABLES;
hive> DESCRIBE passwords;

# Loading data
hive> load data local inpath "/etc/passwd" into table passwords;

# Running a Hive query involving grouping and counts
hive> select col3,count(*) from passwords where col2 > 0 group by col3;

# Cleaning up passwords table
```

```
hive> DROP TABLE passwords;
hive> quit;
```

Testing Hcatalog

Using the HCatalog Command-line API

You can use the following HCatalog command-line to create a table and access table data:

```
# Create a table
$ hcat -e "CREATE TABLE test(key string, value string) ROW FORMAT DELIMITED
 FIELDS TERMINATED BY ','"
OK

# Get the scheme for a table
$ hcat -e "DESC test"
OK
key   string none
value string none
```

Note: Make sure the user is permitted to read the file (e.g., `test_data`) and write the table (e.g., `test`), and the YARN service is running.

Using HCatalog with REST

```
# Get table by using webhcat, you need to change hostname and username to
 appropriate value
$ curl -s 'http://<hostname>:50111/templeton/v1/ddl/database/default/table/
test?user.name=username'
{"columns":[{"name":"key","type":"string"},
{"name":"value","type":"string"}],"database":"default","table":"test"}
```

Using HCatalog with Pig

```
$ pig -useHCatalog
#use HCatLoader to have table schema retrieved automatically
$grunt> A = LOAD 'test' USING org.apache.hcatalog.pig.HCatLoader();
$grunt> DESCRIBE A;
#output
A: {key: chararray,value: chararray}
```

Testing Oozie

Submit Oozie Example Workflows

1. Expand the examples:

   ```
   $ mkdir /tmp/oozie-example
   $ cd /tmp/oozie-example
   $ tar xzf /usr/lib/gphd/oozie/oozie-examples.tar.gz
   ```

2. Change the job properties in the examples. Change the following files:

   ```
   /tmp/oozie-example/examples/apps/map-reduce/job.properties
   /tmp/oozie-example/examples/apps/hive/job.properties
   /tmp/oozie-example/examples/apps/pig/job.properties
   ```

In each file, set the following properties:

```
nameNode=hdfs://<namenode-host>:<namenode-port>
jobTracker=<resource-manager-host>:<resource-manager-port>
```

Use the exact hostname and service port in your cluster.

3. Edit the Oozie `workflow.xml` file as follows:

   Locate the Oozie `workflow.xml` file in the following directory:

   `/tmp/oozie-example/examples/apps/hive`

   Add the NameNode variable as a prefix to all paths. For example:

```
<param>INPUT=${nameNode}/user/${wf:user()}/${examplesRoot}/input-data/
table</param>
<param>OUTPUT=${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/
hive</param>
```

   Also make sure to reference `hive-oozie-site.xml` using the `job-xml` tag in the workflow. The `<job-xml>` element needs to be put inside the `<hive>` element between the `<prepare>` and `<configuration>` elements in the `examples/apps/hive/workflow.xml` file, as shown below:

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="hive-wf">
    <start to="hive-node"/>
    <action name="hive-node">
        <hive xmlns="uri:oozie:hive-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <prepare>
                <delete path="${nameNode}/user/${wf:user()}/
${examplesRoot}/output-data/hive"/>
                <mkdir path="${nameNode}/user/${wf:user()}/
${examplesRoot}/output-data"/>
            </prepare>
            <job-xml>${nameNode}/user/oozie/hive-oozie-site.xml</job-xml>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <script>script.q</script>
            <param>INPUT=${nameNode}/user/${wf:user()}/${examplesRoot}/
input-data/table</param>
            <param>OUTPUT=${nameNode}/user/${wf:user()}/${examplesRoot}/
output-data/hive</param>
        </hive>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Hive failed, error
 message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

4. Put example code onto HDFS:

```
$ hdfs dfs -put examples /user/<username>
```

Where `<username>` is the name of user who issues this command.

5. Submit a MapReduce example workflow:

   a. Submit the workflow:

   ```
   $ oozie job -oozie http://localhost:11000/oozie -config examples/apps/
   map-reduce/job.properties  -run
   job: <oozie-job-id>
   ```

   b. Check the workflow status:

   ```
   $ oozie job -oozie http://localhost:11000/oozie -info <oozie-job-id>
   ```

   Where `<oozie-job-id>` is the same id in the output of the last command.

6. Oozie setup for Hive:

   a. **Remote Metastore Mode (recommended):**

      i. Put the Hive jars into the Tomcat class loader path.

      ii. Make the following change in the `/var/lib/gphd/oozie/tomcat-deployment/conf/catalina.properties` file:

      ```
      common.loader=${catalina.home}/lib,${catalina.home}/lib/*.jar,/var/
      lib/gphd/oozie/*.jar,/usr/lib/gphd/oozie/libtools/*.jar,/usr/lib/
      gphd/oozie/oozie-core/*.jar,/usr/lib/gphd/hadoop/client/*.jar,/usr/
      lib/gphd/hive/lib/*.jar
      ```

      Note: common loader classpath

      Make sure `${catalina.home}/lib,${catalina.home}/lib/*.jar` are at the beginning of the classpath. Keep the jars in the classpath in the following order:

      - Tomcat jars (under `${catalina.home}/lib`)
      - Oozie jars (under `${oozie.home}`, `${oozie.home}/libtools`, `${oozie.home}/oozie-core`)
      - Hadoop jars (under `${hadoop.home}/client/`)
      - Hive jars (under `${hive.home}/lib`)

   b. **Local Metastore Mode:**

   Upload the JDBC driver to Oozie sharelib.

   To enable the local metastore mode, comment out the `hive.metastore.uris` property and verify that Hive still works properly at the command-line. In local metastore mode, Oozie Hive actions do not connect to the Hive Metastore, but instead communicate directly with the database. In this setup, the appropriate JDBC driver (for example, for Postgres) needs to be made available to Hive jobs running within Oozie:

   ```
    sudo -u oozie hdfs dfs -put /usr/lib/gphd/hive/lib/postgresql-
   jdbc.jar /user/oozie/share/lib/hive
   ```

7. Submit the Hive example workflow:

   a. Upload the Hive configuration file onto HDFS:

   ```
   $ sudo -u oozie hdfs dfs -put /etc/gphd/hive/conf/hive-site.xml /user/
   oozie/hive-oozie-site.xml
   ```

   Note: When uploading a Hive configuration file to HDFS, do not use `hive-site.xml` as the file name. This is because the Hive action in Oozie overwrites the `hive-site.xml` file.

In the Oozie workflow file, use `<job-xml>${nameNode}/user/oozie/hive-oozie-site.xml</job-xml>` to refer to the Hive configuration file.

b. Submit the workflow:

```
$ oozie job -oozie http://localhost:11000/oozie -config examples/apps/
hive/job.properties  -run
job: <oozie-job-id>
```

c. Check the workflow status.

```
$ oozie job -oozie http://localhost:11000/oozie -info <oozie-job-id>
```

Where `<oozie-job-id>` is the same id in the output of last command.

8. Submit a Pig example workflow:

a. Submit the workflow:

```
$ oozie job -oozie http://localhost:11000/oozie -config examples/apps/
pig/job.properties  -run
job: <oozie-job-id>
```

b. Check the workflow status.

```
$ oozie job -oozie http://localhost:11000/oozie -info <oozie-job-id>
```

Where `<oozie-job-id>` is the same id in the output of the last command.

Testing Sqoop

Sqoop Client Example

In this example, you use Sqoop to import a MySQL database table into HDFS.

To run this example, in addition to a correctly-installed and configured PHD, you also need to perform the following tasks:

1. Install and run MySQL instance:

```
$ sudo yum -y install mysql
$ sudo service mysqld start
```

2. Install MySQL official JDBC driver and copy `mysql-connector-java.jar` into `/usr/lib/gphd/sqoop/lib`:

```
$ sudo yum -y install mysql-connector-java
$ sudo cp /usr/share/java/mysql-connector-java.jar /usr/lib/gphd/sqoop/lib
```

3. Create MySQL database test and MySQL table student:

```
$ mysql
mysql> use test;
mysql> CREATE TABLE student (id INT PRIMARY KEY, name VARCHAR(100));
mysql> insert into student (id, name) values (1, "John");
mysql> insert into student (id, name) values (2, "Mike");
mysql> insert into student (id, name) values (3, "Tom");
mysql> exit
```

Then run Sqoop to import the table to HDFS:

```
$ sudo -u hdfs hdfs dfs -mkdir -p /tmp
$ sudo -u hdfs hdfs dfs -chmod 777 /tmp
$ sqoop import --connect jdbc:mysql://<mysql_server_host>/test --table
 student --username <username> --target-dir hdfs://<namenode_host>/tmp/
sqoop_output
```

Where:

`<mysql_server_host>` is the host name on which your MySQL instance is running.

`<username>` is the username of the user running this command.

`<namenode_host>` is the host name on which your name node is running.

Testing Flume

Flume Configuration Example

```
$ cat /etc/gphd/flume/conf/flume.conf
agent.sources = r1
agent.sinks = k1
agent.channels = c1

# Describe/configure the source
agent.sources.r1.type = netcat
agent.sources.r1.bind = localhost
agent.sources.r1.port = 44444

# Describe the sink
agent.sinks.k1.type = hdfs
agent.sinks.k1.hdfs.path = hdfs://localhost/user/flume/
agent.sinks.k1.hdfs.fileType = DataStream

# Use a channel which buffers events in memory
agent.channels.c1.type = memory
agent.channels.c1.capacity = 1000
agent.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
agent.sources.r1.channels = c1
agent.sinks.k1.channel = c1
```

Starting/Stopping Flume

Option 1) Using the flume-ng command:

```
$ sudo flume-ng agent -c <config_dir> -f <config_file> -n <agent_name>
```

For example:

```
$ sudo flume-ng agent -c /etc/gphd/flume/conf -f /etc/gphd/flume/conf/
flume.conf -n agent
```

Option 2) Using service commands:

Start/stop the Flume agent by running the following commands:

```
$ sudo service flume-agent start
$ sudo service flume-agent stop
$ sudo service flume-agent status
```

Verifying the Installation

```
$ sudo service flume-agent stop
$ sudo -u hdfs hdfs dfs -mkdir -p /user/flume
$ sudo -u hdfs hdfs dfs -chmod 777 /user/flume
$ sudo service flume-agent start
$ echo hello | nc localhost 44444; sleep 30; sudo -u hdfs hdfs dfs -cat /
user/flume/*
OK
hello
```

Testing Mahout

To test if mahout job is running:

1. Create a sample text file and put it on HDFS.
2. Run a Mahout cat job:

```
hadoop fs -put test_mahout /tmp
(test_mahout is a sample text file)
[gpadmin@centos64-2 ~]$ /usr/bin/mahout cat test_mahout
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
Running on hadoop, using /usr/lib/gphd/hadoop/bin/hadoop and
 HADOOP_CONF_DIR=/etc/gphd/hadoop/conf
MAHOUT-JOB: /usr/lib/gphd/mahout/mahout-examples-0.7-gphd-3.1.0.0-job.jar
Sample mahout test file
14/07/25 11:10:41 INFO driver.MahoutDriver: Program took 6 ms (Minutes:
 1.1666666666666667E-4)
```

Testing PXF

Testing PXF on Hive

Make sure you created a `passwords` table on Hive, which is described in the *Testing Hive* on page 50 section.

Then, go to the HAWQ master node:

```
su - gpadmin
source /usr/lib/gphd/hawq/greenplum_path.sh
psql -p 5432
# gpadmin=# CREATE EXTERNAL TABLE passwords (username text,
 password text, userId text, groupId text, gecos text, home text,
 shell text) LOCATION('pxf://<namenode_host>:50070/passwords?
FRAGMENTER=HiveDataFragmenter&ACCESSOR=HiveAccessor&RESOLVER=HiveResolver')
 format 'custom' (formatter='pxfwritable_import'); ## This is old format.
gpadmin=# CREATE EXTERNAL TABLE passwords (username text, password
 text, userId text, groupId text, gecos text, home text, shell text)
 LOCATION('pxf://{nameservices}/passwords?Profile=hive') format
 'custom' (formatter='pxfwritable_import');
gpadmin=# \d
           List of relations
 Schema |   Name    | Type  |  Owner
--------+-----------+-------+---------
 public | passwords | table | gpadmin
 public | test      | table | gpadmin
(2 rows)
```

```
gpadmin=# select * from passwords;
```

Testing PXF on HBase

```
# a text file has some data
cat hbase-data.txt
create 'hbasestudent', 'rollnum', 'name', 'std'
put 'hbasestudent', 'row1', 'rollnum', '1'
put 'hbasestudent', 'row1', 'name', 'A'
put 'hbasestudent', 'row1', 'std', '3'
put 'hbasestudent', 'row2', 'rollnum', '2'
put 'hbasestudent', 'row2', 'name', 'B'
put 'hbasestudent', 'row2', 'std', '1'
put 'hbasestudent', 'row3', 'rollnum', '3'
put 'hbasestudent', 'row3', 'name', 'C'
put 'hbasestudent', 'row3', 'std', '5'

# Execute it
hbase shell < hbase-data.txt

# in hbase shell, make sure there is the data
scan 'hbasestudent'

su - gpadmin
source /usr/lib/gphd/hawq/greenplum_path.sh
psql -p 5432

#CREATE EXTERNAL TABLE student (recordkey TEXT,
 "rollnum:" TEXT,  "name:" TEXT ,  "std:" TEXT)
 LOCATION ('pxf://    <namenodehost>:50070/hbasestudent?
FRAGMENTER=HBaseDataFragmenter&ACCESSOR=HBaseAccessor&RESOLVER=HBaseResolver'
    ) FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');

For HA cluster
CREATE EXTERNAL TABLE student(recordkey TEXT, "rollnum:" TEXT,  "name:"
 TEXT ,  "std:" TEXT) LOCATION ('pxf://{nameservices}/hbasestudent?
Profile=HBase') FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');

select * from  student;
```

Testing PXF on HDFS

```
cat ranking.txt
Talk Dirty,Jason Derulo,4
All Of Me,John Legend,2
Let It Go,Idina Menzel,5
Happy,Pharrell Williams,1
Dark Horse,Katy Perry,3

hadoop fs -copyFromLocal ranking.txt /tmp

su - gpadmin
source /usr/lib/gphd/hawq/greenplum_path.sh
psql -p 5432

# CREATE EXTERNAL TABLE ranking (song text , artist text, rank
 int) LOCATION ('pxf://<namenodehost>:50070/tmp/ranking.txt?
Fragmenter=HdfsDataFragmenter&ACCESSOR=TextFileAccessor&RESOLVER=TextResolver')
 FORMAT 'TEXT' (DELIMITER = ','); # This is an old way
```

```
CREATE EXTERNAL TABLE ranking (song text , artist text, rank int) LOCATION
 ('pxf://<nameservices>/tmp/ranking.txt?PROFILE=HdfsTextSimple') FORMAT
 'TEXT' (DELIMITER = ',');
# or if you cluster is non-HA
CREATE EXTERNAL TABLE ranking (song text , artist text, rank int) LOCATION
 ('pxf://<namenodehost>:50070/tmp/ranking.txt?PROFILE=HdfsTextSimple')
 FORMAT 'TEXT' (DELIMITER = ',');

select * from ranking order by rank;
```

Post-Install Reference Information

This section provides reference information you might find useful after you've installed PHD.

- *Pivotal HD Directory Layout*
- *SSL Certificates*
- *Cluster Configuration Template Example*

Pivotal HD Directory Layout

The * indicates a designated folder for each Pivotal HD component.

Directory Location	Description
`/usr/lib/gphd/*`	The default `$GPHD_HOME` folder. This is the default parent folder for Pivotal HD components.
`/etc/gphd/*`	The default `$GPHD_CONF` folder. This is the folder for Pivotal HD component configuration files.
`/etc/default/`	The directory used by service scripts to set up the component environment variables.
`/etc/init.d`	The location where a components' Linux Service scripts are stored.
`/var/log/gphd/*`	The default location of the `$GPHD_LOG` directory. The directory for Pivotal HD component logs.
`/var/run/gphd/*`	The location of the any daemon process information for the components.
`/usr/bin`	The folder for the component's command scripts; only sym-links or wrapper scripts are created here.

SSL Certificates

The following table contains information related to SSL certificates:

Port	443	5443
Used by	Apache Default SSL	Command Center UI
Default Certificate Path	/etc/pki/tls/certs/localhost.crt	/usr/local/greenplum-cc/ssl/FQDN.cert
Default Key Path	/etc/pki/tls/private/localhost.key	/usr/local/greenplum-cc/ssl/FQDN.key
Config File	/etc/httpd/conf.d/ssl.conf	/etc/httpd/conf.d/pcc-vhost.conf

Port	443	5443
Post Key Change Step	service httpd restart	service httpd restart
SSL Version	SSLv3 TLSv1.0	SSLv3 TLSv1.0
Compression	No	No
Minimal Encryption Strength	medium encryption (56-bit)	strong encryption (96-bit or more)
ICM Upgrade	No Impact	Check configuration file and key
Support CA Signed Certificates	Yes	Yes

Cluster Configuration Template Example

The `clusterConfig.xml` file contains a default Cluster Configuration template.

The following is an example of the configuration files directory structure:

```
├── clusterConfig.xml
├── hdfs
    ├── core-site.xml
    ├── hadoop-env.sh
    ├── hadoop-metrics2.properties
    ├── hadoop-metrics2.properties
    ├── hadoop-policy.xml
    ├── hdfs-site.xml
    ├── log4j.properties
├── yarn
    ├── container-executor.cfg
    ├── mapred-env.sh
    ├── mapred-queues.xml
    ├── mapred-site.xml
    ├── postex_diagnosis_tests.xml
    ├── yarn-env.sh
    └── yarn-site.xml
└── zookeeper
    └── log4j.properties
    └── zoo.cfg
    └── java.env
├── hbase
    ├── hadoop-metrics.properties
    ├── hbase-env.sh
    ├── hbase-policy.xml
    ├── hbase-site.xml
    ├── jaas.conf
    └── log4j.properties
├── hawq
    └── gpinitsystem_config
├── pig
    ├── log4j.properties
    ├── pig.properties
├── hive
    ├── hive-env.sh
    ├── hive-exec-log4j.properties
    ├── hive-log4j.properties
    ├── hive-site.xml
```

Chapter 6

PHD Pre-Upgrade

This section provides information you'll need, as well as tasks that must be completed, before you upgrade PHD.

Pre-Upgrade Checklist

The following tasks need to be completed before you upgrade PHD.

Each task is explained in more detail in subsequent sections; click the task name to jump to those sections.

Step	Task	Description	Completed
1	*File Locations and Backups*	If you are upgrading PADS, make note of the path to the extracted pre-upgrade PADS tarball. If you don't remember, you can just download it again and untar it. In addition, we recommend that you back up the following before running any upgrade: • Critical data • Configuration files of any services you will be manually reconfiguring after the upgrade.	
2	*Verify Java JDK*	Make sure you are running JDK 1.7. If you are not, download it from Oracle. **Note:** This is a new requirement; prior to PHD 2.0, JDK 1.6 was also supported.	
3	*Compact HBase Tables*	**For upgrades from version 1.1.1 only:** Hbase 0.96 only supports HFile V2 and compacting tables rewrites HFileV1 format to HFile V2.	
4	*Pre-Upgrade 4 - Disable Security on the Cluster (1.1.1 Upgrade Only)* on page 65	**For upgrades from version 1.1.1 only:** You need to disable security before upgrading a PHD 1.1.1 cluster.	
5	Remove GemFire XD BETA	**For upgrades from version 1.1.1 only:**	

Step	Task	Description	Completed
		In PHD 1.1.1, Gemfire XD was BETA. Before upgrading from 1.1.1, you must remove the GemFireXD BETA service, then perform a fresh install of GemFireXD. Data migration from GemfireXD BETA is not supported.	

Additional Tasks:

Task	Description
Sudo Configuration File	If you don't use the automatically-created sudo configuration file, you need to manually add some settings to your own sudo configuration file.

Pre-Upgrade 1 - File Locations and Backup

Before you begin your upgrade, make sure you do the following:

PADS File Location

Make note of the path to the extracted pre-upgrade PADS tarball. If you don't remember, you can just download it again and untar it.

Back Up Data

We recommend you back up any critical data before performing any upgrades.

Back Up Service Configuration Files

Services that were manually installed on an existing cluster are not upgraded by a CLI upgrade. After the PHD upgrade, you need to manually reconfigure these services to work with the upgraded PHD. Back up the configuration files for these services. See *Stack and Tools Reference* for the locations of these configuration files.

Next Task:

Pre-Upgrade 2 - Verify Java JDK on page 63

Pre-Upgrade 2 - Verify Java JDK

Ensure that you are running Oracle JAVA JDK version 1.7 as the default JDK on the Admin node.

Note: This is a new requirement; prior to PHD 2.0, JDK 1.6 was also supported. Instructions below.

Note: Version 1.7 is required; version 1.7u45 is recommended.

Perform the following steps on the Admin node as both `root` and `gpadmin` users:

```
$ /usr/java/default/bin/java -version
```

The output of this command should contain 1.7 (version number) and JavaHotSpot(TM) (Java version). For example:

```
java version "1.7.0_45"
Java(TM) SE Runtime Environment (build 1.7.0_45-b18)
Java HotSpot(TM) 64-Bit Server VM (build 24.45-b08, mixed mode)
```

If you are not running the correct JDK, download a supported version from the Oracle site at *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

Install the JDK on the admin node and add it to alternatives as follows:

```
# /usr/sbin/alternatives --install "/usr/bin/java" "java" "/usr/java/
jdk1.7.0_xx/bin/java" 3
# /usr/sbin/alternatives --install "/usr/bin/javac" "javac" "/usr/java/
jdk1.7.0_xx/bin/javac" 3
# /usr/sbin/alternatives --config java
```

OpenJDK

Make sure you are not running OpenJDK as your default JDK.

If you are running OpenJDK, we recommend you remove it.

To check for all versions of JDK that are running on your system, as `root` run:

```
yum list installed | grep jdk
```

An example output from this command is:

```
java-1.6.0-openjdk.x86_64
java-1.7.0-openjdk.x86_64
jdk.x86_64              2000:1.7.0_45-fcs
```

This indicates that there are three versions of JDK installed, two of them are OpenJDK.

To remove all OpenJDK versions, as `root`, run:

```
yum erase *openjdk*
```

Note: This is a new requirement; prior to PHD 2.0, JDK 1.6 was also supported.

As `gpadmin`, run:

```
$ java -version
java version "1.7.0_15"
Java(TM) SE Runtime Environment (build 1.7.0_15-b03)
Java HotSpot(TM) 64-Bit Server VM (build 23.7-b01, mixed mode)
```

Next Task:

If you are upgrading from 1.1.1, go to *Pre-Upgrade 3 - Compact HBase Tables (1.1.1 Upgrade Only)* on page 64.

If you are upgrading from 2.0.x, you can proceed with *Upgrading PHD 2.0.x to 2.1.0* on page 78.

Pre-Upgrade 3 - Compact HBase Tables (1.1.1 Upgrade Only)

This step is only required if you are upgrading from PHD version 1.1.1.

Before you start your upgrade you need to Compact HBase tables on the existing 0.94 cluster.

For example, to compact table `t1`, log in to the HBase shell, then run:

```
major_compact 't1'
```

Note: HBase 0.96 only supports HFileV2 format and major table compaction rewrites HFileV1 to HfileV2. Skipping this step may lead to data loss.

Next Task:

If you have security enabled on your 1.1.1 cluster, go to *Pre-Upgrade 4 - Disable Security on the Cluster (1.1.1 Upgrade Only)* on page 65.

If you don't have security enabled, you can proceed with *Upgrading PHD 1.1.1 to 2.1.0* on page 90.

Pre-Upgrade 4 - Disable Security on the Cluster (1.1.1 Upgrade Only)

You need to disable security before upgrading a version 1.1.1 cluster.

To disable security:

1. Stop the cluster:

   ```
   [gpadmin]# icm_client stop -l <CLUSTERNAME>
   ```

2. If you have HBase installed and HBase-to-Zookeeper communication is secured (true in most cases), complete the following tasks.

   Tables created while HBase is secure have ACLs set on them that only allow SASL authenticated users to modify them. In order to operate in non-secure mode, you must do the following:

   Note: You can skip these steps if you don't have HBase installed.

   a. Start *just* the Zookeeper service.

      ```
      [gpadmin]# icm_client start -l <CLUSTERNAME> -s zookeeper
      ```

   b. On HBase master:

      i. Run the Zookeeper CLI:

         ```
         [gpadmin]# sudo -u hbase hbase zkcli
         ```

      ii. Check if there are any regions in transition. Output `[]` means there are NO regions in transition at the moment and you don't need to set ACL on this sub znode.

         ```
         [zk:
          node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
          0] ls /hbase/region-in-transition
         []
         ```

         If there are regions in transition, either wait for them to finish (start the cluster again) or set ACL to make them controllable by `world`. Do this for all the regions. For example, if you see a region like 156781230:

         ```
         [zk:
          node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
          1] setAcl /hbase/region-in-tranistion/156781230 world:anyone:cdrwa
         ```

iii. Check if there are unassigned regions. If there are any, set ACL to be controllable by `world`:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 2] ls /hbase/unassigned
[123456789]
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 3] setAcl /hbase/unassigned/123456789 world:anyone:cdrwa
```

iv. Do this for all the tables where ACL is set to anything other than `world:anyone:cdrwa`; otherwise, they won't be readable while security is disabled.

Note: If you're only disabling security temporarily in order to upgrade, and you intend to enable it again after upgrade, you can skip setting ACLs on tables.

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 4] ls /hbase/table
[hbase:meta, hbase:namespace, testtable]
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 5] getAcl /hbase/table/hbase:meta
'world,'anyone
:cdrwa
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 6] getAcl /hbase/table/testtable
'world,'anyone
:r
'sasl,'hbase
:cdrwa
# Here is testtable is not world writable and has SASL enabled.
 If you want to use this table while in non-secure mode, do the
 following.
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 7] setAcl /hbase/table/testtable world:anyone:cdrwa

# Verify ACL has been set
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 8] getAcl /hbase/table/testtable
'world,'anyone
:cdrwa
```

Important: Alternatively, you can also remove the znode `/hbase` or any of its sub-znodes such as `/hbase/table`, as they will be re-created on HBase service restart. Also, this should only be done if HBase-master and HBase-region server were shut down properly and there is no transient state yet to be synced back.

Use this option with *extreme* caution and only if you're having trouble starting HBase service. Careless use may cause data loss.

To remove a znode (e.g. `/hbase/table`), run the following:

```
[zk:
 node2.phddev.local:2181 ,node1.phddev.local:2181,node3.phddev.local:2181(C
 9] rmr /hbase/table
```

v. Quit the Zookeeper CLI on HBase master node. You can now disconnect from HBase master:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 10] quit
```

c. Stop the Zookeeper service from the ICM Admin node.

```
[gpadmin]# icm_client stop -l test -s zookeeper
```

3. You now need to remove security related changes from other service configuration files and scripts. You can use `icm_client reconfigure` for this purpose.

   Note: Make sure it runs successfully on all nodes before proceeding further.

   To use `icm_client reconfigure` to update the configuration file, perform the following steps on the ICM Admin node:

   a. Fetch the current configuration into a directory named `SecureConfiguration`:

   ```
   [gpadmin]# icm_client fetch-configuration -o SecureConfiguration -l
    <CLUSTERNAME>
   ```

   b. Copy `SecureConfiguration` to `NonSecureConfiguration`.

   c. Change to the `NonSecureConfiguration` directory and make the following modifications to disable security-related changes:

   Note: In general, while removing properties, you may ignore and proceed further if the property is already missing, as this could happen because of how the cluster was secured originally. Similarly, while editing properties, if it already has the recommended value, you may safely proceed further.

   i. Remove the following properties from `hdfs/core-site.xml` (if present). Ignore if they're not present, which may be the case in clusters secured without ICM's help.

   hdfs/core-site.xml

   ```
   <property>
     <name>hadoop.security.authentication</name>
     <value>kerberos</value>
   </property>

   <property>
     <name>hadoop.security.authorization</name>
     <value>true</value>
   </property>


   <!-- THE PROPERTY BELOW IS OPTIONAL: IT ENABLES ON WIRE RPC
    ENCRYPTION -->

   <property>
     <name>hadoop.rpc.protection</name>
     <value>privacy</value>
   </property>
   ```

   ii. Remove the following properties from `hdfs/hdfs-site.xml` (if present). Ignore if they're not present, which may be the case in clusters secured without ICM's help.

   hdfs/hdfs-site.xml

   ```
   <property>
   ```

```
  <name>dfs.block.access.token.enable</name>
  <value>true</value>
</property>

<!-- name node secure configuration info -->

<property>
  <name>dfs.namenode.keytab.file</name>
  <value>/etc/security/phd/keytab/hdfs.service.keytab</value>
</property>

<property>
  <name>dfs.namenode.kerberos.principal</name>
  <value>hdfs/_HOST@REALM</value>
</property>

<property>
  <name>dfs.namenode.kerberos.http.principal</name>
  <value>HTTP/_HOST@REALM</value>
</property>

<property>
  <name>dfs.namenode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@REALM</value>
</property>

<!-- (optional) secondary name node secure configuration info -->

<property>
  <name>dfs.secondary.namenode.keytab.file</name>
  <value>/etc/security/phd/keytab/hdfs.service.keytab</value>
</property>

<property>
  <name>dfs.secondary.namenode.kerberos.principal</name>
  <value>hdfs/_HOST@REALM</value>
</property>

<property>
  <name>dfs.secondary.namenode.kerberos.http.principal</name>
  <value>HTTP/_HOST@REALM</value>
</property>

<property>
  <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</
name>
  <value>HTTP/_HOST@REALM</value>
</property>

<!-- If HA is configured -->
<property>
  <name>dfs.journalnode.keytab.file</name>
  <value>/etc/security/phd/keytab/hdfs.keytab</value> <!-- path to
 the HDFS keytab -->
</property>
<property>
  <name>dfs.journalnode.kerberos.principal</name>
  <value>hdfs/_HOST@REALM.COM</value>
</property>
<property>
  <name>dfs.journalnode.kerberos.internal.spnego.principal</name>
  <value>HTTP/_HOST@REALM.COM</value>
</property>
```

```
<property>
  <name>dfs.datanode.kerberos.principal</name>
  <value>hdfs/_HOST@REALM</value>
</property>

<property>
  <name>dfs.datanode.kerberos.http.principal</name>
  <value>HTTP/_HOST@REALM</value>
</property>

<property>
  <name>dfs.datanode.keytab.file</name>
  <value>/etc/security/phd/keytab/hdfs.service.keytab</value>
</property>

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

<property>
  <name>dfs.web.authentication.kerberos.principal</name>
  <value>HTTP/_HOST@REALM</value>
</property>

<property>
  <name>dfs.web.authentication.kerberos.keytab</name>
  <value>/etc/security/phd/keytab/hdfs.service.keytab</value>
</property>

<property>
  <name>dfs.encrypt.data.transfer</name>
  <value>true</value>
</property>

<property>
  <name>dfs.encrypt.data.transfer.algorithm</name>
  <value>rc4</value>
  <description>may be "rc4" or "3des" - 3des has a significant
 performance impact</description>
</property>

<!-- If hive is configured -->
<property>
  <name>hadoop.proxyuser.hive.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hive.groups</name>
  <value>*</value>
</property>


<!-- If oozie is configured -->
<property>
  <name>hadoop.proxyuser.oozie.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.oozie.groups</name>
  <value>*</value>
</property>
```

iii. Edit the following properties in `hdfs/hdfs-site.xml` to the values described below:

hdfs/hdfs-site.xml

```
<!-- For PHD-1.1.1.0 or PHD-1.1.0.0, set this to false -->
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>false</value>
</property>

OR
<!-- For PHD greater than or equal to 2.0, set this to true -->
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>false</value>
</property>

<!-- Following properties should have these values -->
<property>
  <name>dfs.datanode.data.dir.perm</name>
  <value>700</value>
</property>

<property>
  <name>dfs.datanode.address</name>
  <value>0.0.0.0:50010</value>
</property>

<property>
  <name>dfs.datanode.http.address</name>
  <value>0.0.0.0:50075</value>
</property>
```

iv. Edit `hdfs/hadoop-policy.xml`. Search for all instances of `<value>` and replace all instances of `hdfs` with `${HADOOP_HDFS_USER}` and `yarn` with `${HADOOP_YARN_USER}`. Some of the known instances are:

hdfs/hadoop-policy.xml

```
<property>
  <name>security.refresh.usertogroups.mappings.protocol.acl</name>
  <value>${HADOOP_HDFS_USER}</value>
</property>

<property>
  <name>security.refresh.policy.protocol.acl</name>
  <value>${HADOOP_HDFS_USER}</value>
</property>

<property>
  <name>security.qjournal.service.protocol.acl</name>
  <value>${HADOOP_HDFS_USER}</value>
</property>

<!-- YARN Protocols -->
<property>
  <name>security.resourcetracker.protocol.acl</name>
  <value>${HADOOP_YARN_USER}</value>
</property>

<property>
  <name>security.admin.protocol.acl</name>
  <value>${HADOOP_YARN_USER}</value>
</property>
```

v. Remove the following properties from `yarn/yarn-site.xml` (if present). Ignore if they're not present, which may be the case in clusters secured without ICM's help.

yarn/yarn-site.xml

```
<property>
  <name>yarn.resourcemanager.principal</name>
  <value>yarn/_HOST@REALM</value>
</property>

<property>
  <name>yarn.resourcemanager.keytab</name>
  <value>/etc/security/phd/keytab/yarn.service.keytab</value>
</property>

<property>
  <name>yarn.nodemanager.principal</name>
  <value>yarn/_HOST@REALM</value>
</property>

<property>
  <name>yarn.nodemanager.keytab</name>
  <value>/etc/security/phd/keytab/yarn.service.keytab</value>
</property>

<property>
  <name>yarn.nodemanager.container-executor.class</name>

 <value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</
value>
</property>

<property>
  <name>yarn.nodemanager.linux-container-executor.group</name>
  <value>yarn</value>
</property>

<property>
  <name>yarn.web-proxy.keytab</name>
  <value>/etc/security/phd/keytab/yarn.service.keytab</value>
</property>

<property>
  <name>yarn.web-proxy.principal</name>
  <value>yarn/_HOST@REALM</value>
</property>
```

vi. Remove the following properties from `yarn/mapred-site.xml`:

yarn/mapred-site.xml

```
<property>
  <name>mapreduce.jobhistory.keytab</name>
  <value>/etc/security/phd/keytab/mapred.service.keytab</value>
</property>

<property>
  <name>mapreduce.jobhistory.principal</name>
  <value>mapred/_HOST@REALM</value>
</property>
```

vii. Edit `yarn/container-executor.cfg` as follows:

yarn/container-executor.cfg

```
#configured value of yarn.nodemanager.linux-container-executor.group
yarn.nodemanager.linux-container-executor.group=
#comma separated list of users who can not run applications
banned.users=
#Prevent other super-users
min.user.id=1000
```

viii. Remove the following lines from `yarn/container-executor.cfg`:

yarn/container-executor.cfg

```
yarn.nodemanager.local-dirs=/data/1/yarn/nm-local-dir
yarn.nodemanager.log-dirs=/data/1/yarn/userlogs
```

ix. Remove the following lines from `zookeeper/zoo.cfg`:

zookeeper/zoo.cfg

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000

kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

x. For PHD-2.0.0.0 and higher, edit `zookeeper/java.env` to remove – `Djava.security.auth.login.config=/etc/gphd/zookeeper/conf/jaas.conf` from `JVMFLAGS`.

zookeeper/java.env

```
export JVMFLAGS="-Xmx2048m"
```

xi. Remove the following properties from `hbase/hbase-site.xml`:

hbase/hbase-site.xml

```
<property>
  <name>hbase.security.authentication</name>
  <value>kerberos</value>
</property>

<property>
  <name>hbase.security.authorization</name>
  <value>true</value>
</property>

<property>
  <name>hbase.rpc.engine</name>
  <value>org.apache.hadoop.hbase.security.access.AccessController</
value>
</property>

<property>
  <name>hbase.coprocessor.master.classes</name>
  <value>org.apache.hadoop.hbase.security.access.AccessController,
 org.apache.hadoop.hbase.security.token.TokenProvider</value>
</property>

<property>
  <name>hbase.coprocessor.region.classes</name>
  <value>org.apache.hadoop.hbase.security.access.AccessController,
 org.apache.hadoop.hbase.security.token.TokenProvider</value>
```

```
</property>

<!-- HBase secure region server configuration -->
<property>
  <name>hbase.regionserver.kerberos.principal</name>
  <value>hbase/_HOST@REALM</value>
</property>

<property>
  <name>hbase.regionserver.keytab.file</name>
  <value>/etc/security/phd/keytab/hbase.service.keytab</value>
</property>

<!-- HBase secure master configuration -->
<property>
  <name>hbase.master.kerberos.principal</name>
  <value>hbase/_HOST@REALM</value>
</property>

<property>
  <name>hbase.master.keytab.file</name>
  <value>/etc/security/phd/keytab/hbase.service.keytab</value>
</property>

<property>
  <name>hbase.rest.keytab.file</name>
  <value>path-to-rest-users-keytab</value>
</property>

<property>
  <name>hbase.rest.kerberos.principal</name>
  <value>rest-users-principal-name</value>
</property>
```

xii. Remove the following line from `hbase/hbase-env.sh`:

hbase/hbase-env.sh

```
export HBASE_OPTS="$HBASE_OPTS -Djava.security.auth.login.config=/
etc/gphd/hbase/conf/jaas.conf"
```

xiii Remove the following properties from `hive/hive-site.xml`:

hive/hive-site.xml

```
<property>
  <name>hive.server2.authentication</name>
  <value>KERBEROS</value>
</property>

<property>
  <name>hive.server2.authentication.kerberos.principal</name>
  <value>hive/_HOST@REALM</value>
</property>

<property>
  <name>hive.server2.authentication.kerberos.keytab</name>
  <value>/etc/security/phd/keytab/hive.keytab</value>
</property>

<property>
  <name>hive.server2.enable.impersonation</name>
  <value>true</value>
</property>
```

```
<property>
  <name>hive.server2.enable.doAs</name>
  <value>true</value>
</property>

<property>
  <name>hive.metastore.sasl.enabled</name>
  <value>true</value>
  <description>If true, the metastore thrift interface will be
 secured with SASL. Clients
   must authenticate with Kerberos.</description>
</property>

<property>
  <name>hive.security.authorization.enabled</name>
  <value>true</value>
  <description>enable or disable the hive client authorization</
description>
</property>

<property>
  <name>hive.security.authorization.createtable.owner.grants</name>
  <value>ALL</value>
  <description>the privileges automatically granted to the owner
 whenever a table gets created.
   An example like "select,drop" will grant select and drop privilege
 to the owner of the table.
   You may change this value if you desire lower privileges on
 create.</description>
</property>

<property>
  <name>hive.metastore.kerberos.keytab.file</name>
  <value>/etc/security/phd/keytab/hive.keytab</value>
  <description>The path to the Kerberos Keytab file containing the
 metastore thrift
   server's service principal.</description>
</property>

<property>
  <name>hive.metastore.kerberos.principal</name>
  <value>hive-metastore/_HOST@REALM</value>
  <description>The service principal for the
 metastore thrift server. The special string _HOST will be replaced
automatically with the correct host name.</description>
</property>
```

xiv For HAWQ: If present, remove the following properties from `hawq/hdfs-client.xml`:

If these properties are not present, you must manually remove these XML tags on HAWQ nodes after running `icm_client reconfigure` to disable security on the cluster.

hawq/hdfs-client.xml

```
<property>
  <name>hadoop.security.authentication</name>
  <value>kerberos</value>
</property>

<property>
  <name>dfs.namenode.kerberos.principal</name>
  <value>HDFS_NAMENODE_PRINCIPAL</value>
```

```
</property>
```

xv. For HAWQ: Remove the following lines from `hawq/gpinitsystem_config`:

hawq/gpinitsystem_config

```
KERBEROS_KEYFILE=/path/to/keytab/file
ENABLE_SECURE_FILESYSTEM=on
```

d. Run `icm_client reconfigure` using the `NonSecureConfiguration` directory you just modifed to push these changes to cluster nodes:

```
[gpadmin]# icm_client reconfigure -l <CLUSTERNAME> -c
 NonSecureConfiguration
```

4. With the cluster services still stopped, **comment** the following lines (if present) in `/etc/default/hadoop-hdfs-datanode` on **ALL** DataNodes.

/etc/default/hadoop-hdfs-datanode (on DataNode)

```
# secure operation stuff -- comment the following lines, if present and
 not commented. Ignore if a property is missing.
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_LOG_DIR=${HADOOP_LOG_DIR}/hdfs
export HADOOP_SECURE_DN_PID_DIR=${HADOOP_PID_DIR}
```

5. For PHD-1.1.1.0 and lower, remove `/etc/gphd/zookeeper/conf/java.env` from all zookeeper-server nodes (if present). We recommend that you back up the file before removing it.

6. Remove security from any manually-installed service, following the reverse of the instructions to enable them.

7. Start the cluster:

```
[gpadmin]# icm_client start -l <CLUSTERNAME>
```

8. If HAWQ is configured, do the following on the HAWQ master as `gpadmin`:

a. Source the HAWQ path:

```
source /usr/local/hawq/greenplum_path.sh
```

b. If not already running, start HAWQ by running:

```
/etc/init.d/hawq start
```

c. Specify that security is not enabled by running:

```
gpconfig --masteronly -c enable_secure_filesystem -v off
```

At this point, security should be disabled and you may run test commands to validate data is still accessible in non-secure mode.

Next Task:

None. You can proceed with *Upgrading PHD 1.1.1 to 2.1.0* on page 90.

sudo Configuration File

The sudo configurations in `/etc/sudoers.d/gpadmin` are used by the `gpadmin` user to perform deployments and upgrades. This sudo configuration file is automatically created as part of the `preparehosts` command that is run during deployments and upgrades.

If you don't use the configuration files under /etc/sudoers.d due to your site security policy, you need to add the following sudo settings to your sudo configuration file to allow the gpadmin user to perform deployment and upgrade tasks. This needs to be done before attempting to deploy or upgrade.

```
####
Defaults:root,%gpadmin !requiretty


Defaults:root,%gpadmin secure_path += /sbin:/bin:/usr/sbin:/usr/bin:/usr/
local/bin


## Networking
Cmnd_Alias PCC_SYSTEM_NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /
sbin/dhclient, /sbin/iptables


### Installation and management of software
Cmnd_Alias PCC_SYSTEM_SOFTWARE = /bin/cp, /bin/mv, /bin/mkdir, /bin/grep, /
usr/bin/tee, /sbin/sysctl, /bin/chmod, /bin/chown, /bin/rpm, /usr/bin/
yum, /usr/bin/puppet, /usr/bin/createrepo, /usr/bin/ssh-keygen, /usr/sbin/
setenforce, /usr/sbin/useradd, /usr/sbin/ntpdate, /usr/bin/test, /usr/sbin/
alternatives, /usr/sbin/authconfig


### Commands with specific params
Cmnd_Alias PCC_COMMANDS_SPECIFIC_PARAMS = /bin/rm -rf /etc/gphd/*, /bin/rm
 -rf /etc/security/phd/*, /bin/rm -rf /usr/lib/gphd/*, /bin/rm -rf /var/lib/
gphd/*, /bin/rm -rf /var/log/gphd/*, /bin/rm -rf /tmp/.massh-gpadmin, /bin/
rm -rf ~gpadmin/*


### Services
Cmnd_Alias PCC_SYSTEM_SERVICES = /sbin/service, /sbin/chkconfig


### PCC specific services
Cmnd_Alias PCC_SERVICES = /etc/init.d/hadoop-hdfs-namenode, /etc/init.d/
hadoop-hdfs-datanode, /etc/init.d/hadoop-hdfs-secondarynamenode, /etc/
init.d/hadoop-yarn-resourcemanager, /etc/init.d/hadoop-yarn-nodemanager, /
etc/init.d/hadoop-mapreduce-historyserver, /etc/init.d/zookeeper-server, /
etc/init.d/hbase-master, /etc/init.d/hbase-regionserver, /etc/init.d/hive-
server, /etc/init.d/hive-metastore, /etc/init.d/postgresql, /etc/init.d/
hawq, /etc/init.d/uss-namenode, /home/gpadmin/jdk, /etc/init.d/hadoop-hdfs-
journalnode, /etc/init.d/hadoop-hdfs-zkfc, /etc/init.d/nodeagent, /etc/
init.d/zabbix-agent, /etc/init.d/pxf-service


### ICM Preparehost scripts
Cmnd_Alias PCC_PREPAREHOST_CMDS = /tmp/gphdgmr/addHawqConfigs.py


%gpadmin ALL=(root) NOPASSWD: PCC_SYSTEM_SOFTWARE, PCC_SYSTEM_SERVICES,
 PCC_SYSTEM_NETWORKING, PCC_SERVICES, PCC_COMMANDS_SPECIFIC_PARAMS,
 PCC_PREPAREHOST_CMDS


%gpadmin ALL=(hadoop,hdfs,mapred,yarn,hbase,hive,zookeeper,postgres)
 NOPASSWD:ALL
```

```
#####
```

Chapter 7

Upgrading PHD 2.0.x to 2.1.0

This section describes how to upgrade Pivotal HD using Pivotal Command Center's command line interface (CLI).

2.0.x to 2.1.0 - Upgrade Checklist

Note: Before you start your upgrade; make sure you have met all the upgrade prerequisites (see *Pre-Upgrade Checklist* on page 62).

The table below briefly describes the tasks you must complete to upgrade PHD.

Each task is explained in more detail in the next section (*2.0.x to 2.1.0 - Upgrade Instructions* on page 80).

Step	Task	Details	Completed

1	Verify the state of your cluster	Make sure your cluster is healthy and in a consistent state:
		1. Use the PCC UI to make sure there are no services down or running with errors.
		On one of the HDFS nodes, as `gpadmin`, run:
		`$ sudo -u hdfs hdfs dfsadmin -report`
		Check the output for issues.
		2. Check the health of the file system by running:
		`$ sudo -u hdfs hdfs fsck /`	

2	Stop Services	**Stop HAWQ** (if applicable):
		`$ /etc/init.d/hawq stop`
		(See*Managing HAWQ* on page 143 for details.)
		Stop all PHD services - As `gpadmin` run:
		`$ icm_client stop -l <CLUSTER NAME>`
		(See *Managing a PHD Cluster* on page 112f or details.)
		Stop PCC - As `root` run:
		`$ service commander stop`	

Step	Task	Details	Completed

		Stop GemFire XD locator - On the locator node, as `root` run:
		`$ gfxd locator stop -dir=<PATH_TO_LOCATOR_DIR>`
		Stop GemFire XD servers - On each GemFire server node, as `root` run:
		`$ gfxd server stop -dir=<PATH_TO_SERVER_DIR>`	

3	Import and Upgrade PCC	Untar the new PCC package, then run (as `root`):
		`$ ./install`
		Change the user to `gpadmin` for the rest of the upgrade.	

4	Import HAWQ (PADS)	`$ icm_client import -`
		`s <PATH_TO_EXTRACTED_PADS_TARBALL>`	

5	Import PRTS	`$ icm_client import -`
		`s <PATH_TO_EXTRACTED_PRTS_TARBALL>`	

6	Import PHD	`$ icm_client import -`
		`s PATH_TO_EXTRACTED_PHD_TARBALL>`	

7	Edit the Configuration File	Fetch the existing configuration file to a user specified directory (`~/upgraded_conf` in this example):
		`$ icm_client fetch-upgrade-configuration -v`
		` PHD-2.1.0.0 -l test -o ~/upgraded_conf`
		For the PXF service and Security values, edit as described in *Edit Configuration File*.	

8	Upgrade HAWQ (PADS)	`$ icm_client upgrade -l <CLUSTERNAME> -s pads -`
		`o <PATH_TO_EXTRACTED_OLD_PADS_TARBALL>`
		` -n <PATH_TO_EXTRACTED_NEW_PADS_TARBALL>`	

9	Upgrade PRTS	`$ icm_client upgrade -l <CLUSTERNAME> -s prts`	
10	Upgrade PHD	`$ icm_client upgrade -l <CLUSTERNAME> -s phd`	

Step	Task	Details	Completed
11	PXF with GemFire XD	Add `'/usr/lib/gphd/gfxd/lib/gemfirexd.jar'` on a new line to the `ClusterConfigDir/pxf/pxf-public.classpath` file.	
12	Reconfigure the Cluster	`$ icm_client reconfigure -l test -c ~/upgraded_conf`	
13	Restart the Cluster	`$ icm_client restart -l <CLUSTERNAME>`	
14	Restart HAWQ	`$ /etc/init.d/hawq start`	
15	Reconfigure Manually-Installed Services	Services that were installed manually on an existing cluster are not upgraded by a CLI upgrade. After upgrade, you need to manually reconfigure these services to work with the upgraded PHD.	
16	Move HAWQ Filespace to HA-enabled HDFS	For HA clusters: If you are using HAWQ, you need to move the HAWQ filespace to HA-enabled HDFS. See *2.0.x to 2.1.0 - Move HAWQ Filespace to HA-enabled HDFS* on page 84 for details.	

2.0.x to 2.1.0 - Upgrade Instructions

Note: Before you start your upgrade; make sure you have met all the upgrade prerequisites (see *Pre-Upgrade Checklist* on page 62).

Follow the instructions below to upgrade PHD 2.0.1 to PHD 2.1.0:

1. **Verify the current state of the cluster**

   a. Using the Pivotal Command Center user interface, check to see if any services are down. If any service is down or is running with errors, address those issues before upgrading.

   b. On one of the HDFS nodes, as `gpadmin`, run:

   ```
   sudo -u hdfs hdfs dfsadmin -report
   ```

   An example of the output is below.

   Make sure that there are no:

   - `Under replicated blocks`, `Blocks with corrupt replicas`, or `Missing blocks`.
   - Dead or decommissioned nodes:
     - If you have decommissioned Data Nodes, removed then from the cluster using the `icm_client remove-slaves` command (see *Shrinking a Cluster*). You can always add them back after you have completed the upgrade procedure (see *Expanding a Cluster*).
     - If you have dead Data Nodes, either remove then or bring them back up.

   Example dfsadmin Report

   ```
   sudo -u hdfs hdfs dfsadmin -report
   Configured Capacity: 93657587712 (87.23 GB)
   ```

```
Present Capacity: 81391808512 (75.80 GB)
DFS Remaining: 81391706112 (75.80 GB)
DFS Used: 102400 (100 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
-------------------------------------------------
Datanodes available: 1 (1 total, 0 dead)
Live datanodes:
Name: 192.168.2.203:50010 (rhel64-3.localdomain)
Hostname: rhel64-3.localdomain
Decommission Status : Normal
Configured Capacity: 93657587712 (87.23 GB)
DFS Used: 102400 (100 KB)
Non DFS Used: 12265779200 (11.42 GB)
DFS Remaining: 81391706112 (75.80 GB)
DFS Used%: 0.00%
DFS Remaining%: 86.90%
Last contact: Fri Apr 25 18:39:22 UTC 2014
```

c. Run `fsck` and ensure that the filesystem is healthy; for example there are no corrupt files. An example of the output is below.

Example fsck Output

```
sudo -u hdfs hdfs fsck /
Connecting to namenode via http://rhel64-3:50070
FSCK started by hdfs (auth:SIMPLE) from /192.168.2.202 for path / at Fri
 Apr 25 20:56:52 UTC 2014
...Status: HEALTHY
Total size: 366 B
Total dirs: 20
Total files: 3
Total symlinks: 0
Total blocks (validated): 3 (avg. block size 122 B)
Minimally replicated blocks: 3 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Fri Apr 25 20:56:52 UTC 2014 in 211 milliseconds

The filesystem under path '/' is HEALTHY
```

Important: If you cannot get a cluster into a healthy state, contact Pivotal Support before continuing with your upgrade.

2. **Stop Services:**

a. Stop HAWQ. As `gpadmin`, on the HAWQ master run:`$ /etc/init.d/hawq stop`

b. Stop PHD services. As `gpadmin`, run:`$ icm_client stop -l <CLUSTERNAME>`

c. Stop PCC. As `root`, run:

```
$ service commander stop
```

d. Stop GemFire XD locator. On the locator node, as `root` run:

```
$ gfxd locator stop -dir=<path to locator dir>
```

e. Stop GemFire XD servers. On each GemFire server node, as `root` run:

```
$ gfxd server stop -dir=<path to server dir>
```

3. **Import and upgrade PCC:**

a. Download the new PCC file from *https://network.pivotal.io/*.

b. Copy the new PCC tar file to your installation directory on the admin node, for example:

```
$ scp ./PCC-2.3.x.version.build.os.x86_64.tar.gz host:/root/phd/
```

c. Login as `root` and untar to that directory:

```
$ cd /root/phd
$ tar --no-same-owner -zxvf PCC-2.3.x.version.build.os.x86_64.tar.gz
```

d. As `root`, run the PCC installation script from the directory where it is installed:

```
$ ./install
```

Note: There is no need to specify that this is an upgrade; the install utility (./install) detects whether it is a fresh install or an upgrade.

Important: The rest of the upgrade procedure is performed by the `gpadmin` user. Switch to that user now.

4. **Import new HAWQ package:**

a. Download and extract the new PADS (HAWQ) package from *https://network.pivotal.io/*.

b. Run:

```
$ icm_client import -s <PATH_TO_EXTRACTED_PADS_TARBALL>
```

5. **Import new PRTS package:**

a. Download and extract the new PRTS (GemFire XD) package from *Pivotal Network*.

b. Run:

```
$ icm_client import -s <PATH_TO_EXTRACTED_PRTS_TARBALL>
```

6. **Import new PHD package:**

a. Download and extract the new PHD package from *Pivotal Network*.

b. Run:

```
$ icm_client import -s <PATH_TO_EXTRACTED_PHD_TARBALL>
```

7. **Edit the Configuration File:**

a. Retrieve the auto-generated cluster configuration for the PHD package you are upgrading to by running `icm_client fetch-upgrade-configuration` command. You only need to provide the PHD version for the PHD package you are upgrading to as the value for the `-v` option as shown in the example. This cluster configuration would be used to reconfigure the cluster after the package upgrades are successful (see *Upgrade HAWQ* and *Upgrade PHD*).

For example, as `gpadmin`, run:

```
$ mkdir ~/upgraded_conf
$ icm_client fetch-upgrade-configuration -v PHD-2.1.0.0 -l <CLUSTERNAME>
 -o ~/upgraded_conf
```

b. This step is only required if `gpxf` was a configured service in the existing cluster configuration.

Make the following changes to `clusterConfig.xml` in your newly created `upgraded_conf` directory:

i. Remove `gpxf` from the `<services>` list.

ii. Add `pxf` to the `<services>` list.

iii. Add `pxf-service` role to `<hostRoleMapping>`. Colocate the `pxf-service` role with namenode and datanode.

```
<pxf>
   <pxf-service></pxf-service>
</pxf>
```

iv. Delete the `gpxf` directory from `upgraded_conf` directory:

```
$ rm -rf ~/upgraded_conf/gpxf
```

v. Add the new PXF template to `upgraded_conf`. You can do this by fetching the new template and copying the `pxf` directory from the template.

For example, as `gpadmin`, run:

```
$ mkdir ~/new_template
$ icm_client fetch-template -o ~/new_template
$ cp -r ~/new_template/pxf upgraded_conf
```

c. Specify security. The configuration file already has a `<securityEnabled>` parameter. Set this to either **True** or **False**. If **True**, follow the steps for configuring security after this step.

8. Upgrade HAWQ:

Note: This section is only applicable if you installed Pivotal ADS (HAWQ) using PHD's CLI; if you installed Pivotal ADS manually, refer to the *HAWQ Installation and Upgrade Guide* for manual upgrade instructions.

a. To upgrade PADS (HAWQ), as `gpadmin`, run:

```
$ icm_client upgrade -l <CLUSTERNAME> -s pads
 -o <PATH_TO_EXTRACTED_OLD_ADS_TARBALL> -n
 <PATH_TO_EXTRACTED_NEW_ADS_TARBALL>
```

b. **Optional:** You can delete the old HAWQ rpm file by running:

```
$ yum erase <HAWQ_OLD_RPM_NAME>
```

9. Upgrade PRTS:

To upgrade PRTS (GemFire XD), as `gpadmin` run:

```
$ icm_client upgrade -l <CLUSTERNAME> -s prts
```

10. Upgrade PHD:

If your cluster is configured with HAWQ, make sure you complete upgrading Pivotal ADS (see previous step), before proceeding with Pivotal HD upgrade. To upgrade PHD, as `gpadmin`, run:

```
$ icm_client upgrade -l <CLUSTERNAME> -s phd
```

This upgrades the PHD stack on all cluster nodes.

11. PXF with GFXD:

If you have PXF using GFXD as a data source, add `'/usr/lib/gphd/gfxd/lib/gemfirexd.jar'` on a new line to `ClusterConfigDir/pxf/pxf-public.classpath`.

12. **Reconfigure the cluster:**

    Reconfigure your cluster with the new upgraded configuration:

    As `gpadmin`, run:

    ```
    $ icm_client reconfigure -l <CLUSTERNAME> -c ~/upgraded_conf
    ```

13. **Restart the cluster:**

    As `gpadmin`, run:

    ```
    $ icm_client restart -l <CLUSTERNAME>
    ```

14. **Restart HAWQ:**

    As `gpadmin`, run:

    ```
    $ /etc/init.d/hawq start
    ```

15. **Reconfigure Manually-Installed Services:**

    Services that were installed manually on an existing cluster are not upgraded by a CLI upgrade. After the PHD upgrade, you need to manually reconfigure these services to work with the upgraded PHD. See the *PHD Stack and Tool Reference Guide* for details.

    Note: Backing up the configuration files for these services is a prerequisite for this upgrade procedure. See the *PHD Stack and Tools Reference Guide* for the locations of these configuration files.

Next Task:

If you are using HAWQ in an HA environment, you need to move the HAWQ filespace to HA-enabled HDFS, as described in the next step, *2.0.x to 2.1.0 - Move HAWQ Filespace to HA-enabled HDFS* on page 84.

If you are not using HAWQ in an HA environment, your cluster is now upgraded. At this point, you should check to see if all your services are running and your data is intact. *Running PHD Sample Programs* on page 44 provides instructions for testing the various services.

2.0.x to 2.1.0 - Move HAWQ Filespace to HA-enabled HDFS

For HAWQ in an HA environment, you need to perform the following tasks to complete your upgrade.

As HAWQ was initialized, post-upgrade, on a non-HA HDFS, you now need to move the HAWQ filespace to HA-enabled HDFS, as follows:

Collecting Information about the Target Filespace

A default filespace named `dfs_system` exists in the `pg_filespace` catalog and the parameter `pg_filespace_entry` contains detailed information for each filespace.

1. Use the following SQL query to gather information about the filespace located on HDFS:

    ```
    SELECT
        fsname, fsedbid, fselocation
    FROM
        pg_filespace as sp, pg_filespace_entry as entry, pg_filesystem as fs
    ```

```
WHERE
    sp.fsfsys = fs.oid and fs.fsysname = 'hdfs' and sp.oid =
 entry.fsefsoid
ORDER BY
    entry.fsedbid;
```

The sample output is as follows:

```
fsname   | fsedbid |             fselocation
------------+---------+--------------------------------------
 dfs_system |       1 | /data/hawq-kerberos/dfs/gpseg-1
 dfs_system |       2 | hdfs://mdw:9000/hawq-security/gpseg0
 dfs_system |       3 | hdfs://mdw:9000/hawq-security/gpseg1
 dfs_system |       4 | hdfs://mdw:9000/hawq-security/gpseg2
 dfs_system |       5 | hdfs://mdw:9000/hawq-security/gpseg3
 dfs_system |       6 | hdfs://mdw:9000/hawq-security/gpseg4
 dfs_system |       7 | hdfs://mdw:9000/hawq-security/gpseg5
(7 rows)
```

The output can contain the following:

- Master instance path information.
- Standby master instance path information, if the standby master is configured (not in this example).
- HDFS paths that share the same prefix for segment instances.

2. To enable HA HDFS, you need the segment location comprising the filespace name and the common prefix of segment HDFS paths. The segment location is formatted like a URL. The sample output displays the segment location, `hdfs://mdw:9000/hawq-security`. Where `mdw:9000` is the Namenode host and RPC port, you must replace it with your HA HDFS cluster service ID to get the new segment location. For example:

```
hdfs://phdcluster/hawq-security
Filespace Name: dfs_system
New segment location: hdfs://phdcluster/hawq-security
```

Note: To move the filespace location to a segment location that is different from the old segment location, you must move the data to new path on HDFS.

For example, move the filespace from `hdfs://phdcluster/hawq-security` to `hdfs://phdcluster/hawq/another/path`.

Stop the HAWQ Cluster and Back Up the Catalog

To enable HA HDFS, you are changing the HAWQ catalog and persistent tables. You cannot preform transactions while persistent tables are being updated. Therefore, before you stop the HAWQ Cluster, Pivotal recommends that you back up the catalog. This is to ensure that you do not lose data due to a hardware failure or during an operation (such as killing the HAWQ process).

1. Disconnect all workload connections.
2. Issue a checkpoint.
3. Shutdown the HAWQ cluster.
4. Define `$MASTER_DATA_DIRECTORY` to point to the `MASTER_DATA_DIRECTORY` path:

```
export MASTER_DATA_DIRECTORY=<MASTER_DIRECTORY>/gpseg-1
```

For example:

```
export MASTER_DATA_DIRECTORY=/data0/master/gpseg-1/gps
```

5. Copy the master data directory:

```
cp -r $MASTER_DATA_DIRECTORY /catalog/backup/location
```

Move the Filespace Location

HAWQ provides the command line tool, `gpfilespace`, to move the location of the filespace.

Run the following command line to move a filespace location:

```
gpfilespace --movefilespace default --location=hdfs://phdcluster/hawq-
security
```

Note:

1. If the target filespace is not the default filespace, replace the default in the command line with the actual filespace name.
2. Replace `hdfs://phdcluster/hawq-security` with the new segment location.

Important: Errors while moving the location of the filespace:

A non-fatal error can occur if you provide invalid input or if you have not stopped HAWQ before attempting a filespace location change. Check that you have followed the instructions from the beginning, or correct the input error before you re-run `gpfilespace`.

Fatal errors can occur due to hardware failure or if you fail to kill a HAWQ process before attempting a filespace location change. When a fatal error occurs, you will see the message, `"PLEASE RESTORE MASTER DATA DIRECTORY"` in the output. If this occurs, shut down the database and restore the `$MASTER_DATA_DIRECTORY`.

Configure ${GPHOME}/etc/hdfs-client.xml

Configure the `hdfs-client.xml` file. See the *HAWQ Installation and Upgrade Guide* for more information.

Reinitialize the Standby Master

The standby master catalog is rendered invalid during the move, and needs to be reinitialized. If you did not have a standby master configured, you can skip this task.

```
gpstart -a                                #start HAWQ cluster
gpinitstandby -r                          #remove standby master
gpinitstandby -s <standby host name>      #initialize a standby master
```

Next Task:

None. Your upgrade is now complete. At this point, you should check to see if all your services are running and your data is intact. *Running PHD Sample Programs* on page 44 provides instructions for testing the various services.

2.0.x to 2.1.0 - Upgrade Reference Information

- *Upgrade Syntax*
- *Changed Configuration Parameters and Files*

Upgrade Syntax

For reference, the complete syntax for the `upgrade` command is as follows:

```
[gpadmin]# icm_client upgrade --help
Usage: /usr/bin/icm_client upgrade [options]

Options:
  -h, --help            show this help message and exit
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        the name of the cluster on which the operation is
                        performed
  -x, --noscanhosts     Do not verify cluster nodes.
  -s STACK, --stackname=STACK
                        stack to upgrade (phd or pads)
  -v VERSION, --version=VERSION
                        PHD Stack version, default is PHD-2.0.0.0 Stack
  -o OLDDIR, --old=OLDDIR
                        (Required for only for pads/hawq upgrade) Old PADS
                        Directory
  -n NEWDIR, --new=NEWDIR
                        (Required for only for pads/hawq upgrade) New PADS
                        Directory
  -p, --nopreparehosts  Do not prepare hosts as part of deploying the
 cluster
  -j JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM (Ex: jdk-
                        7u15-linux-x64.rpm). Ignored if -p is specified
  -t, --ntp             Synchronize system clocks using NTP. Optionally
 takes
                        NTP server as argument. Defaults to pool.ntp.org
                        (requires external network access). Ignored if -p is
                        specified
  -d, --selinuxoff      Disable SELinux. Ignored if -p is specified
  -i, --iptablesoff     Disable iptables. Ignored if -p is specified
  -y SYSCONFIGDIR, --sysconf=SYSCONFIGDIR
                        [Only if HAWQ is part of the deploy] Directory
                        location of the custom conf files (sysctl.conf and
                        limits.conf) which will be appended to
                        /etc/sysctl.conf and /etc/limits.conf on slave
 nodes.
                        Default: /usr/lib/gphd/gphdmgr/hawq_sys_config/.
                        Ignored if -p is specified
```

Changed Configuration Parameters and Files

The following information is provided solely as reference material; you do not need to make any changes to your configuration files beyond those you have already completed.

The following configuration parameters were changed in PHD 2.1 as described below:

hbase-site.xml

New Parameters

The following parameters have been added to `hbase-site.xml`:

Name	Default Value	Description
hbase.bulkload.staging.dir	/apps/hbase/staging	Directory in the default filesystem, owned by the hbase user, and has permissions (-rwx--x--x, 711)

Chapter 8

Upgrading PHD 1.1.1 to 2.1.0

This section describes how to upgrade Pivotal HD using Pivotal Command Center's command line interface (CLI).

1.1.1 to 2.1.0 - Upgrade Checklist

Note: Before you start your upgrade; make sure you have met all the upgrade prerequisites (see *Pre-Upgrade Checklist* on page 62).

The table below briefly describes the tasks you must complete to upgrade PHD.

Each task is explained in more detail in the next section (*1.1.1 to 2.1.0 - Upgrade Instructions* on page 93).

Step	Task	Details	Completed

1	Verify the state of your cluster	Make sure your cluster is healthy and in a consistent state:
		1. Use the PCC UI to make sure there are no services down or running with errors.
		2. On one of the HDFS nodes, as `gpadmin`, run:
		`$ sudo -u hdfs hdfs dfsadmin -report`
		Check the output for issues.
		3. Check the health of the file system by running:
		`$ sudo -u hdfs hdfs fsck /`	

2	Back up Hive metastore	Log in to the machine running the Hive metastore database, then run: `pg_dump -U hive -p 10432 metastore > hive_metastore_1.backup`	
3	Disable High Availability	If High Availability is enabled, disable it before you begin your upgrade. See *Disabling High Availability on a Cluster* for instructions. To complete this step, run the following SQL command: `psql -U postgres -p 10432 gphdmgr -c "UPDATE cluster_properties SET property_value='false' WHERE <cluster_id>=2 AND property_name='cluster.nn.isHAEnabled';"`	

Step	Task	Details	Completed
4	Revert to Non-Secure	If security is enabled, disable it before you begin your upgrade. See *Pre-Upgrade 4 - Disable Security on the Cluster (1.1.1 Upgrade Only)* on page 65 for instructions.	
5	Co-locate Hive server with Name Node	**For PXF: Co-locate your Hive server and Name Node (if applicable).** Your upgrade from PHD 1.1.1 to PHD 2.1 with PXF installed will fail if your hive server is not co-located with your Name Node. To co-locate them, add the `hive.noarch` package on the Name Node and copy `hive-site.xml` from hive-server node to the Name Node machine. **Note:** As part of *PHD Pre-Upgrade* on page 62, you should have already removed the GemFire service.	

6	Remove Standby HAWQ master	Remove the Standby HAWQ master:
		1. Source the `greenplum_path.sh`:
		`$ source /usr/local/hawq/greenplum_path.sh`
		2. Then, as `gpadmin`, run:
		`$ gpinitstandby -r`	

7	Stop Services	Stop HAWQ (if applicable):
		`$ /etc/init.d/hawq stop`
		(See*Managing HAWQ* on page 143 for details.)
		As `gpadmin`, stop all PHD services:
		`$ icm_client stop -l <CLUSTERNAME>`
		(See *Managing a PHD Cluster* on page 112 for details.)
		As `root`, stop PCC:
		`$ service commander stop`	

8	Import and Upgrade PCC	Untar the new PCC package, then run (as `root`):
		`$ ./install`
		Change the user to `gpadmin` for the rest of the upgrade.	

9	CLI Self-Upgrade	`$ icm_client self-upgrade`	

10	Import HAWQ (PADS)	`$ icm_client import -`
		`s <PATH_TO_EXTRACTED_PADS_TARBALL>`	

11	Upgrade HAWQ (PADS)	To upgrade HAWQ, run:
		`$ icm_client upgrade -l <CLUSTERNAME> -s pads -`
		`o <PATH_TO_EXTRACTED_OLD_PADS_TARBALL >`	

Step	Task	Details	Completed
		`-n <PATH_TO_EXTRACTED_NEW_PADS_TARBALL>`	

12	Import PRTS (GemFire XD)	`$ icm_client import -`
		`s <PATH_TO_EXTRACTED_PRTS_TARBALL>`	

13	Upgrade PRTS (GemFire XD)	`$ icm_client upgrade -l <CLUSTERNAME> -s prts`	

14	Import PHD	`$ icm_client import -`
		`s <PATH_TO_EXTRACTED_PHD_TARBALL>`	

15	Upgrade PHD	PHD 2.0.1 and above requires Oracle JDK 1.7:
		1. Get the JDK RPM (for example: `jdk-7u45-linux-x64.rpm`).
		2. Include the RPM in the upgrade command as shown below, so that the `upgrade` command can deploy it to the cluster nodes:
		`$ icm_client upgrade -l <CLUSTERNAME> -s phd -j`
		` ~/jdk-7u45-linux-x64.rpm`	

16	PXF with GemFire as data source	If you have PXF using GemFire XD (GFXD) as a data source, add `'/usr/lib/gphd/gfxd/lib/gemfirexd.jar'` on a new line to `ClusterConfigDir/pxf/pxf-public.classpath`.	
17	Upgrade Configuration Files	**1.** Synchronize configuration files. **2.** Reconfigure the cluster. **Note:** Do not add any security or HA-specific configuration parameters/values at this time, wait until you have completed the upgrade.	
18	Upgrade HDFS	**1.** Back up Name Node data. **2.** Run `HdfsUpgrader.py` with appropriate options (see *Upgrading PHD 2.0.x to 2.1.0* on page 78 for details).	
19	Restart the Cluster	`$ icm_client restart -l <CLUSTERNAME>`	
20	Post-Upgrade HAWQ	**Note:** If you have MADlib dependencies, see the *HAWQ Installation and Upgrade Guide* for instructions for upgrading MADlib. To migrate HAWQ data, on the HAWQ master node, run: `gpmigrator <OLD_HAWQHOME_PATH> <NEW_HAWQHOME_PATH>` Note that this command also starts HAWQ.	

Step	Task	Details	Completed
		Reinitialize the HAWQ Standby Master: `$ gpinitstandby -s <STANDBY_HOSTNAME>`	
21	Finalize HDFS Upgrade	Run the `FinalizeHDFS` command.	

22	Finalize HBase Upgrade	**1.** Check for HFileV1 data (not supported after upgrade):
		`$ sudo -u hbase hbase upgrade -check`
		2. Make sure Zookeeper and HDFS are running, but HBase is stopped.
		3. Run the HBase upgrade:
		`$ sudo -u hbase hbase upgrade -execute`	

23	Reconfigure Manually Installed Services	Services that were manually installed on an existing cluster are not upgraded by a CLI upgrade. After upgrade, you need to manually reconfigure these services to work with the upgraded PHD.	
24	Re-enable High Availability	See *High Availability* on page 125 for details.	
25	Re-secure Cluster	See *Security/Kerberos Authentication* on page 136 for details.	
26	Move HAWQ Filespace to HA-enabled HDFS	For HA clusters: For HAWQ, you need to move the HAWQ filespace to HA-enabled HDFS. See *1.1.1 to 2.1.0 - Move HAWQ Filespace to HA-enabled HDFS* on page 101 for details.	
27	Add GFXD service	For GemFire XD: once you have upgraded PHD, you need to to reconfigure the cluster to add the GFXD service. See *Adding/ Removing Services* on page 116 for details.	

1.1.1 to 2.1.0 - Upgrade Instructions

Note: Before you start your upgrade; make sure you have met all the upgrade prerequisites (see *Pre-Upgrade Checklist* on page 62).

Follow the instructions below to upgrade your PHD system from 1.1.1 to 2.1.0:

1. **Verify the current state of the cluster:**
   a. Using the Pivotal Command Center user interface, check to see if any services are down. If any service is down or is running with errors, address those issues before upgrading.
   b. On one of the HDFS nodes, as `gpadmin`, run:

   ```
   sudo -u hdfs hdfs dfsadmin -report
   ```

   An example of the output is below.

Make sure that there are no:

- `Under replicated blocks`, `Blocks with corrupt replicas`, or `Missing blocks`.
- Dead or decommissioned nodes:
  - If you have decommissioned Data Nodes, removed then from the cluster using the `icm_client remove-slaves` command (see *Shrinking a Cluster*). You can always add them back after you have completed the upgrade procedure (see *Expanding a Cluster*).
  - If you have dead Data Nodes, either remove then or bring them back up.

Example dfsadmin Report

```
sudo -u hdfs hdfs dfsadmin -report
Configured Capacity: 93657587712 (87.23 GB)
Present Capacity: 81391808512 (75.80 GB)
DFS Remaining: 81391706112 (75.80 GB)
DFS Used: 102400 (100 KB)
DFS Used%: 0.00%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
-------------------------------------------------
Datanodes available: 1 (1 total, 0 dead)
Live datanodes:
Name: 192.168.2.203:50010 (rhel64-3.localdomain)
Hostname: rhel64-3.localdomain
Decommission Status : Normal
Configured Capacity: 93657587712 (87.23 GB)
DFS Used: 102400 (100 KB)
Non DFS Used: 12265779200 (11.42 GB)
DFS Remaining: 81391706112 (75.80 GB)
DFS Used%: 0.00%
DFS Remaining%: 86.90%
Last contact: Fri Apr 25 18:39:22 UTC 2014
```

c. Run `fsck` and ensure that the filesystem is healthy; for example, there are no corrupt files. An example of the output is below.

Example fsck Output

```
sudo -u hdfs hdfs fsck /
Connecting to namenode via http://rhel64-3:50070
FSCK started by hdfs (auth:SIMPLE) from /192.168.2.202 for path / at Fri
 Apr 25 20:56:52 UTC 2014
...Status: HEALTHY
Total size: 366 B
Total dirs: 20
Total files: 3
Total symlinks: 0
Total blocks (validated): 3 (avg. block size 122 B)
Minimally replicated blocks: 3 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Fri Apr 25 20:56:52 UTC 2014 in 211 milliseconds
```

```
The filesystem under path '/' is HEALTHY
```

Important: If you cannot get a cluster into a healthy state, contact Pivotal Support before continuing with your upgrade.

2. **Back up the Hive metastore:**

   Hive does not provide rollback options, so we recommend that you take a snapshot of the metastore DB before starting the upgrade:

   a. As `gpadmin`, log in to the machine running the Hive metastore database.

   b. Use the following command to backup the metastore database. It will back up the metastore database to file `hive_metastore_1.backup`:

   ```
   pg_dump -U hive -p 10432 metastore > hive_metastore_1.backup
   ```

3. **Disable High Availability (if applicable):**

   You cannot upgrade a version 1.1.1 cluster with High Availability enabled. Revert your cluster to non-HA before proceeding with an upgrade. See *Disabling HA* for details. To complete this step for upgrades, run the following SQL command:

   ```
   psql -U postgres -p 10432 gphdmgr
   -c "UPDATE cluster_properties SET property_value='false'
   WHERE cluster_id=<cluster_id> AND property_name='cluster.nn.isHAEnabled';"
   ```

   Where: `<cluster_id>>` is the id of your cluster.

4. **Revert to Non-Secure (if applicable):**

   You cannot upgrade a version 1.1.1 cluster with security enabled. Revert your cluster to non-secure before proceeding with an upgrade. See *Pre-Upgrade 4 - Disable Security on the Cluster (1.1.1 Upgrade Only)* on page 65 for details.

5. **For PXF, co-locate your Hive server and Name Node (if applicable):**

   Your upgrade from PHD 1.1.1 to PHD 2.1 with PXF installed will fail if your Hive server is not co-located with your Name Node. To co-locate these, add the `hive.noarch` package on the Name Node and copy `hive-site.xml` from the hive-server node to the Name Node machine.

6. **Remove HAWQ Standby Master:**

   If you have a HAWQ Standby Master, you need to remove it before you start the upgrade. As `gpadmin`, do the following:

   a. Source the `greenplum_path.sh` file:

   ```
   $ source /usr/local/hawq/greenplum_path.sh
   ```

   b. Remove the HAWQ Standby Master by running:

   ```
   $ gpinitstandby -r
   ```

   For more details, see the *HAWQ Installation and Upgrade Guide*.

7. **Stop Services:**

   a. As `gpadmin`, stop HAWQ on the HAWQ master:

   ```
   $ /etc/init.d/hawq stop
   ```

   b. As `gpadmin`, stop all PHD services:

   ```
   $ icm_client stop -l <CLUSTERNAME>
   ```

c. As `root`, stop PCC:

```
$ service commander stop
```

8. Import and upgrade PCC:

a. Download the new PCC file from *Pivotal Network*.

b. Copy the new PCC tarball file to your installation directory on the admin node. For example:

```
$ scp ./PCC-2.3.x.version.build.os.x86_64.tar.gz host:/root/phd/
```

c. Log in as `root` and untar to that directory:

```
$ cd /root/phd
$ tar --no-same-owner -zxvf PCC-2.3.x.version.build.os.x86_64.tar.gz
```

d. As `root`, run the PCC installation script from the directory where it is installed:

```
$ ./install
```

Note: There is no need to specify that this is an upgrade; the install utility (`./install`) detects whether it is a fresh install or an upgrade.

Important: The rest of the upgrade procedure is performed by the `gpadmin` user. Switch to that user now.

9. CLI Self-Upgrade:

As `gpadmin`, run the following command to upgrade the CLI:

```
$ icm_client self-upgrade
```

Note: This command may return very quickly. This does not indicate any problems and you can continue with the upgrade.

10.Import new HAWQ package:

a. Download and extract the new PADS (HAWQ) package from *Pivotal Network*.

b. Run:

```
$ icm_client import -s <PATH_TO_EXTRACTED_PADS_TARBALL>
```

11.Upgrade HAWQ:

Important: This section is only applicable if you installed Pivotal ADS (HAWQ) using PHD's CLI; if you installed Pivotal ADS manually, refer to the *HAWQ Installation and Upgrade Guide* for manual upgrade instructions.

a. To upgrade PADS (HAWQ), as `gpadmin`, run:

```
$ icm_client upgrade -l <CLUSTERNAME> -s pads -
o <PATH_TO_EXTRACTED_OLD_ADS_TARBALL>
-n <PATH_TO_EXTRACTED_NEW_ADS_TARBALL>
```

b. Optional: You can delete the old HAWQ rpm file by running:

```
$ yum erase <HAWQ_OLD_RPM_NAME>
```

12.Import new PRTS (for GemFire XD) package:

a. Download and extract the new PRTS package from *Pivotal Network*.

b. Run:

```
$ icm_client import -s <PATH_TO_EXTRACTED_PRTS_TARBALL>
```

13.Upgrade PRTS (for GemFire XD):

As `gpadmin`, run:

```
$ icm_client upgrade -l <CLUSTERNAME> -s prts
```

14.Import new PHD package:

a. Download and extract the new PHD package from *Pivotal Network*.

b. Run:

```
$ icm_client import -s <PATH_TO_EXTRACTED_PHD_TARBALL>
```

15.Upgrade PHD:

If your cluster is configured with HAWQ, make sure you complete upgrading Pivotal ADS (Upgrade HAWQ step, above), before proceeding with Pivotal HD upgrade.

PHD 2.x requires Oracle JDK 1.7. If you are already running JDK 1.7, proceed with the PHD Upgrade, step b, below. If you need to upgrade to JDK 1.7, first complete step a, below.

a. Import JDK:

JDK 1.7 running on the Admin node is a prerequisite. This step is to import a downloaded JDK package that will be deployed across the cluster.

i. Download a supported JDK package from *http://www.oracle.com/technetwork/java/javase/downloads/index.html*. PHD expects an RPM package; for example: `jdk-7u45-linux-x64.rpm`.

ii. Import the downloaded JDK package to the cluster nodes. As `gpadmin`, run:

```
$ icm_client import -r <PATH_TO_JDK>
```

Note: If you have manually installed UnlimitedJCEPolicy files prior to upgrading your JDK, you will need to re-install them post upgrade.

b. To upgrade PHD, as `gpadmin`, run:

```
$ icm_client upgrade -l <CLUSTERNAME> -s phd
```

If you need to upgrade to JDK 1.7, include the imported JDK rpm in the upgrade command (for example: `jdk-7u45-linux-x64.rpm`) so that the `upgrade` command can deploy it to the cluster nodes:

```
$ icm_client upgrade -l <CLUSTERNAME> -s phd -j ~/jdk-7u45-linux-
x64.rpm
```

This upgrades the PHD stack on all cluster nodes.

Note: All upgrade steps, including post-upgrade configuration steps described below, should be completed before you re-enable HA or security on a cluster.

16.PXF with GemFire XD:

If you have PXF using GemFire XD as a data source, add:

```
'/usr/lib/gphd/gfxd/lib/gemfirexd.jar'
```

on a new line to:

```
ClusterConfigDir/pxf/pxf-public.classpath
```

17. **Upgrade Configuration Files:**

    After upgrading the PHD stack, you need to upgrade your cluster configuration files:

    a. Fetch the new templates that come with the upgraded stack by running `icm_client fetch-template`. For example:

    ```
    icm_client fetch-template -o ~/newTemplate
    ```

    `newTemplate` is the new template for the upgraded stack without any user customizations.

    b. Retrieve the existing configuration from the database by running `icm_client fetch-configuration`. For example:

    ```
    icm_client fetch-configuration -o ~/origConfiguration -l <CLUSTERNAME>
    ```

    `origConfiguration` is based on user-customized template from a previous installation.

    c. Identify the changes between the configurations by running the `diff` command. For example:

    ```
    diff -ruBw newTemplate/ origConfiguration/
    ```

    Then apply those changes to the `newTemplate` you retrieved.

    Tip: To simplify the process of merging the existing PHD configuration with the `newTemplate`, follow these steps:

    i. Overwrite `clusterConfig.xml` in `newTemplate` with the one from the `origConfiguration` directory:

    ```
    cp ~/origConfiguration/clusterConfig.xml  ~/newTemplate/
    clusterConfig.xml
    ```

    ii. Change the value of `<gphdStackVer>` to `PHD-2.1.0.0` in the `~/newTemplate/clusterConfig.xml` file.

    iii. If you have explicitly modified any properties from PHD services configuration files, such as `hdfs/hdfs-site.xml`, `yarn/yarn-site.xml`, etc., then make the corresponding changes to these configuration files under the `~newTemplate/` directory.

    d. This step is only required if `gpxf` was a configured service in the existing cluster configuration.

    Make the following changes to `clusterConfig.xml` in your `newTemplate` directory:

    i. Remove `gpxf` from the `<services>` list.

    ii. Add `pxf` to the `<services>` list.

    iii. Add `pxf-service` role to `<hostRoleMapping>`.

    iv. Colocate the `pxf-service` role with namenode and datanode.

    ```
    <pxf>
    <pxf-service></pxf-service>
    </pxf>
    ```

    v. Delete the `gpxf` directory from `newTemplate` directory:

    ```
    $ rm -rf newTemplate/gpxf
    ```

    vi. Add the new PXF template to `newTemplate`. You can do this by fetching the new template and copying the `pxf` directory from the template.

For example, as gpadmin, run:

```
$ mkdir new_template
$ icm_client fetch-template -o new_template
$ cp -r new_template/pxf newTemplate
```

e. Change the `<gphdVersion>` field to `PHD- 2.1.0.0`.

f. Upgrade services by specifying the cluster configuration directory as `~/newTemplate` with your updated contents:

```
icm_client reconfigure -c ~/newTemplate -l <CLUSTERNAME> -f
```

18.Upgrade HDFS:

Note: If you are performing the upgrade on an EMC Data Computing Appliance (DCA), you need to make sure that the gpadmin user has read access to each of the subdirectories of the NameNode name directories. The location of the NameNode name directories is specified in the value of the `dfs.namenode.name.dir` property in `/etc/gphd/hadoop/conf/hdfs-site.xml` on the NameNode.

For example, if /`data/nn/dfs/name` is the NameNode directory, then the `gpadmin` user must have read access to `data`, `nn` , `dfs` and `name` directories.

As `gpadmin`, on the Admin node, do the following:

a. Backup the NameNode metadata by running:

```
/usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py -
l <CLUSTERNAME> -o backupNNMetadata -s 2.0.5_alpha_gphd_2_1_1_0 -t
 2.2.0_gphd_3_1_0_0
```

b. Run the NameNode upgrade by running:

```
/usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py
 -l <CLUSTERNAME> -o nnupgrade -s 2.0.5_alpha_gphd_2_1_1_0 -t
 2.2.0_gphd_3_1_0_0
```

c. Run the DataNode upgrade by running:

```
/usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py
 -l <CLUSTERNAME> -o dnupgrade -s 2.0.5_alpha_gphd_2_1_1_0 -t
 2.2.0_gphd_3_1_0_0
```

19.Restart the cluster:

As `gpadmin`, run:

```
$ icm_client restart -l <CLUSTERNAME>
```

20.Post-Upgrade HAWQ:

a. Before you perform the following tasks, if you have MADlib dependencies, see the *HAWQ Installation and Upgrade Guide* for instructions for upgrading MADlib.

b. On the HAWQ master node, as `gpadmin`, run the following commands to migrate data:

```
su - gpadmin
source /usr/lib/gphd/hawq/greenplum_path.sh
gpmigrator <old_HAWQHOME_path> <new_HAWQHOME_path> # Look into ls -laF /
usr/local and find the old and new homes.

# For example:
gpmigrator /usr/local/hawq-1.1.4.0/ /usr/local/hawq-1.2.1.0/ -d /data1/
master/gpseg-1
```

Note: The `gpmigrator` command also starts HAWQ.

If you encounter errors migrating HAWQ data, see the *HAWQ Administrator Guide* for help.

c. If you were utilizing a standby HAWQ master, you should have removed it before the upgrade. It should now be reinitialized.

On the HAWQ master, as `gpadmin`, run:

```
$ gpinitstandby -s <standby_hostname>
```

For more details about these commands, see the *HAWQ Installation and Upgrade Guide*.

21.Finalize the HDFS upgrade:

Before you continue, you should run a few tests to make sure your data upgrade was successful, and then you can run `finalizeUpgrade`.

Once you have confirmed your cluster is working as expected, run the following command to finalize the upgrade process:

```
/usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py -
l <CLUSTERNAME> -o finalizeUpgrade -s 2.0.5_alpha_gphd_2_1_1_0 -t
 2.2.0_gphd_3_1_0_0
```

Note: HBase master will not start unless the HBase upgrade is finalized. Please ensure HDFS upgrade is finalized before finalizing HBase upgrade.

22.Finalize HBase Upgrade:

a. Check for any HFileV1 data (only HFileV2 is supported after upgrade to HBase 0.96). On the hbase-master, run:

```
$ sudo -u hbase hbase upgrade -check
```

If the return is `Count of HFileV1:0`, continue with the upgrade.

Note: As part of the prerequisites, you should have already compacted all the tables on the existing HBase cluster; this will have overwritten any HFile V1 data to HFile V2 format.

b. Make sure Zookeeper and HDFS are running, but HBase is stopped.

c. Run:

```
$ sudo -u hbase hbase upgrade -execute
```

23.Reconfigure Manually-Installed Services:

Services that were installed manually on an existing cluster are not upgraded by a CLI upgrade. After the PHD upgrade, you need to manually reconfigure these services to work with the upgraded PHD. See the *PHD Stack and Tool Reference* for details.

Note: Backing up the configuration files for these services is a prerequisite for this upgrade procedure. See the *PHD Stack and Tools Reference* for the locations of these configuration files.

24.Re-enable High Availability:

See *High Availability* on page 125 for details. Note that for fresh installations of PHD 2.1, high availability is enabled by default. For upgrades however, you will have to re-enable high availability.

25.Re-secure the Cluster:

See *Security/Kerberos Authentication* on page 136 for details.If you are not using HAWQ in a HA environment, your cluster should now be upgraded.

Next Task:

- **For HA Clusters:**

For HAWQ in an HA environment, you need to move the HAWQ filespace to HA-enabled HDFS, as described in the next section, *1.1.1 to 2.1.0 - Move HAWQ Filespace to HA-enabled HDFS* on page 101.

- **For GemFire XD:**

  Once you have upgraded PHD, you need to to reconfigure the cluster to add the GemFire XD (GFXD) service. See *Adding/Removing Services* on page 116 for details.

Your cluster should now be upgraded. At this point, you should check to see if all your services are running and your data is intact. *Running PHD Sample Programs* on page 44 provides instructions for testing the various services.

1.1.1 to 2.1.0 - Move HAWQ Filespace to HA-enabled HDFS

For HAWQ in an HA environment, you need to perform the following to complete your upgrade.

As HAWQ was initialized, post-upgrade, on a non-HA HDFS, you now need to move the HAWQ filespace to HA-enabled HDFS, as follows:

Collecting Information about the Target Filespace

A default filespace named `dfs_system` exists in the `pg_filespace` catalog and the parameter `pg_filespace_entry` contains detailed information for each filespace.

1. Use the following SQL query to gather information about the filespace located on HDFS:

```
SELECT
    fsname, fsedbid, fselocation
FROM
    pg_filespace as sp, pg_filespace_entry as entry, pg_filesystem as fs
WHERE
    sp.fsfsys = fs.oid and fs.fsysname = 'hdfs' and sp.oid =
 entry.fsefsoid
ORDER BY
    entry.fsedbid;
```

The sample output is as follows:

```
fsname    | fsedbid |             fselocation
------------+---------+--------------------------------------
 dfs_system |       1 | /data/hawq-kerberos/dfs/gpseg-1
 dfs_system |       2 | hdfs://mdw:9000/hawq-security/gpseg0
 dfs_system |       3 | hdfs://mdw:9000/hawq-security/gpseg1
 dfs_system |       4 | hdfs://mdw:9000/hawq-security/gpseg2
 dfs_system |       5 | hdfs://mdw:9000/hawq-security/gpseg3
 dfs_system |       6 | hdfs://mdw:9000/hawq-security/gpseg4
 dfs_system |       7 | hdfs://mdw:9000/hawq-security/gpseg5
(7 rows)
```

The output can contain the following:

- Master instance path information.
- Standby master instance path information, if the standby master is configured (not in this example).
- HDFS paths that share the same prefix for segment instances.

2. To enable HA HDFS, you need the segment location comprising the filespace name and the common prefix of segment HDFS paths. The segment location is formatted like a URL. The sample output

displays the segment location, `hdfs://mdw:9000/hawq-security.` Where `mdw:9000` is the Namenode host and RPC port, you must replace it with your HA HDFS cluster service ID to get the new segment location. For example:

```
hdfs://phdcluster/hawq-security
Filespace Name: dfs_system
New segment location: hdfs://phdcluster/hawq-security
```

Note: To move the filespace location to a segment location that is different from the old segment location, you must move the data to new path on HDFS.

For example, move the filespace from `hdfs://phdcluster/hawq-security` to `hdfs://phdcluster/hawq/another/path`.

Stop the HAWQ Cluster and Back Up the Catalog

To enable HA HDFS, you are changing the HAWQ catalog and persistent tables. You cannot preform transactions while persistent tables are being updated. Therefore, before you stop the HAWQ Cluster, Pivotal recommends that you back up the catalog. This is to ensure that you do not lose data due to a hardware failure or during an operation (such as killing the HAWQ process).

1. Disconnect all workload connections.
2. Issue a checkpoint.
3. Shutdown the HAWQ cluster.
4. Define `$MASTER_DATA_DIRECTORY` to point to the `MASTER_DATA_DIRECTORY` path:

```
export MASTER_DATA_DIRECTORY=<MASTER_DIRECTORY>/gpseg-1
```

For example:

```
export MASTER_DATA_DIRECTORY=/data0/master/gpseg-1/gps
```

5. Copy the master data directory:

```
cp -r $MASTER_DATA_DIRECTORY /catalog/backup/location
```

Move the Filespace Location

HAWQ provides the command line tool, `gpfilespace`, to move the location of the filespace.

1. Run the following command line to move a filespace location:

```
gpfilespace --movefilespace default --location=hdfs://phdcluster/hawq-
security
```

Note:

a. If the target filespace is not the default filespace, replace the default in the command line with the actual filespace name.
b. Replace `hdfs://phdcluster/hawq-security` with the new segment location.

Important: Errors while moving the location of the filespace:

A non-fatal error can occur if you provide invalid input or if you have not stopped HAWQ before attempting a filespace location change. Check that you have followed the instructions from the beginning, or correct the input error before you re-run `gpfilespace`.

Fatal errors can occur due to hardware failure or if you fail to kill a HAWQ process before attempting a filespace location change. When a fatal error occurs, you will see the message, "PLEASE RESTORE MASTER DATA DIRECTORY" in the output. If this occurs, shut down the database and restore the $MASTER_DATA_DIRECTORY.

Configure ${GPHOME}/etc/hdfs-client.xml

Configure the hdfs-client.xml file. See the *HAWQ Installation and Upgrade Guide* for information.

Reinitialize the Standby Master

The standby master catalog is rendered invalid during the move, and needs to be reinitialized. If you did not have a standby master configured, you can skip this task.

```
gpstart -a                              #start HAWQ cluster
gpinitstandby -r                        #remove standby master
gpinitstandby -s <standby host name>    #initialize a standby master
```

Next Task:

None. Your upgrade is now complete. At this point, you should check to see if all your services are running and your data is intact. *Running PHD Sample Programs* on page 44 provides instructions for testing the various services.

1.1.1 to 2.1.0 - Upgrade Reference Information

- *Upgrade Syntax* on page 103
- *Changed Configuration Parameters and Files* on page 104

Upgrade Syntax

For reference, the complete syntax for the upgrade command is as follows:

```
[gpadmin]# icm_client upgrade --help
Usage: /usr/bin/icm_client upgrade [options]

Options:
  -h, --help            show this help message and exit
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        the name of the cluster on which the operation is
                        performed
  -x, --noscanhosts     Do not verify cluster nodes.
  -s STACK, --stackname=STACK
                        stack to upgrade (phd or pads)
  -v VERSION, --version=VERSION
                        PHD Stack version, default is PHD-2.0.0.0 Stack
  -o OLDDIR, --old=OLDDIR
                        (Required for only for pads/hawq upgrade) Old PADS
                        Directory
  -n NEWDIR, --new=NEWDIR
                        (Required for only for pads/hawq upgrade) New PADS
                        Directory
  -p, --nopreparehosts  Do not prepare hosts as part of deploying the
 cluster
  -j JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM (Ex: jdk-
                        7u15-linux-x64.rpm). Ignored if -p is specified
  -t, --ntp             Synchronize system clocks using NTP. Optionally
 takes
                        NTP server as argument. Defaults to pool.ntp.org
```

```
                       (requires external network access). Ignored if -p is
                       specified
 -d, --selinuxoff      Disable SELinux. Ignored if -p is specified
 -i, --iptablesoff     Disable iptables. Ignored if -p is specified
 -y SYSCONFIGDIR, --sysconf=SYSCONFIGDIR
                       [Only if HAWQ is part of the deploy] Directory
                       location of the custom conf files (sysctl.conf and
                       limits.conf) which will be appended to
                       /etc/sysctl.conf and /etc/limits.conf on slave
nodes.
                       Default: /usr/lib/gphd/gphdmgr/hawq_sys_config/.
                       Ignored if -p is specified
```

Changed Configuration Parameters and Files

The following information is provided solely as reference material; you do not need to make any changes to your configuration files beyond those you have already completed.

The following configuration parameters were changed in PHD 2.0 as described below:

core-site.xml

Removed Parameters

The following parameters have been removed from `core-site.xml`:

Name	Default	Notes
kfs.stream-buffer-size	4096	KFS is no longer supported, see*https://issues.apache.org/jira/browse/HADOOP-8886*
mapred.outdir.resolverClass	org.apache.hadoop.mapreduce.DefaultPathResolver	
kfs.client-write-packet-size	65536	KFS is no longer supported, see *https://issues.apache.org/jira/browse/HADOOP-8886*
kfs.blocksize	67108864	KFS is no longer supported, see *https://issues.apache.org/jira/browse/HADOOP-8886*
kfs.bytes-per-checksum	512	KFS is no longer supported, see *https://issues.apache.org/jira/browse/HADOOP-8886*
kfs.replication	3	KFS is no longer supported, see *https://issues.apache.org/jira/browse/HADOOP-8886*

New Parameters

The following parameters have been added to `core-site.xml`:

Name	Default
fs.client.resolve.remote.symlinks	true
nfs3.server.port	2049
nfs3.mountd.port	4242

Name	Default
hadoop.security.group.mapping.ldap.directory.search.timeout	10000
ipc.client.fallback-to-simple-auth-allowed	false

yarn-site.xml

Changed Defaults

The following parameters in `yarn-site.xml` have new default values:

Name	Old Value	New Value
yarn.nodemanager.aux-services	mapreduce.shuffle	mapreduce_shuffle

New Names

The following parameters in `yarn-site.xml` have new names:

Old Name	New Name	Default Value
yarn.resourcemanager.fs.rm-state-store.uri	yarn.resourcemanager.fs.state-store.uri	${hadoop.tmp.dir}/yarn/system/rmstore
yarn.nodemanager.resource.cpu-cores	yarn.nodemanager.resource.cpu-vcores	8 See *https://issues.apache.org/jira/browse/YARN-782*
yarn.nodemanager.aux-services.mapreduce.shuffle.class	yarn.nodemanager.aux-services.mapreduce_shuffle.class	org.apache.hadoop.mapred.ShuffleHandler
yarn.nodemanager.heartbeat.interval-ms	yarn.resourcemanager.nodemanagers.heartbeat-interval-ms	1000
yarn.resourcemanager.am.max-retries	yarn.resourcemanager.am.max-attempts	1->2

Removed Parameters

The following parameters have been removed from `yarn-site.xml`:

Name	Default Value	Note
net.topology.with.nodegroup	false	Introduced by hve patch. Will be added when the patch is added again to hadoop 2.2.0
yarn.dynamic.resource.memory.minimum.mb	0	Introduced by hve patch. Will be added when the patch is added again to hadoop 2.2.0
yarn.dynamic.resource.vcores.maximum	-1	Introduced by hve patch. Will be added when the patch is added again to hadoop 2.2.0
yarn.dynamic.resource.enable	true	Introduced by hve patch.

Name	Default Value	Note
		Will be added when the patch is added again to hadoop 2.2.0
yarn.dynamic.resource.memory.maximum.mb	-1	Introduced by hve patch. Will be added when the patch is added again to hadoop 2.2.0
yarn.dynamic.resource.vcores.minimum	0	Introduced by hve patch. Will be added when the patch is added again to hadoop 2.2.0
yarn.nodemanager.vcores-pcores-ratio	2	See *https://issues.apache.org/jira/browse/YARN-782*

New Parameters

The following parameters have been added to `yarn-site.xml`:

Name	Default Value
yarn.resourcemanager.connect.retry-interval.ms	30000
yarn.resourcemanager.connect.max-wait.ms	900000
yarn.client.nodemanager-client-async.thread-pool-max-size	500
yarn.resourcemanager.hostname	0.0.0.0
yarn.resourcemanager.scheduler.monitor.enable	false
yarn.http.policy	HTTP_ONLY
yarn.nodemanager.hostname	0.0.0.0
yarn.client.max-nodemanagers-proxies	500
yarn.resourcemanager.webapp.https.address	0.0.0.0:8090
yarn.nodemanager.resourcemanager.connect.wait.secs	900
yarn.client.app-submission.poll-interval	1000
yarn.resourcemanager.scheduler.monitor.policies	org.apache.hadoop.yarn.server.resourcemanager.monitor.capacit
yarn.nodemanager.local-cache.max-files-per-directory	8192
yarn.nodemanager.resourcemanager.connect.retry_interval.secs	30

hdfs-site.xml

Changed Defaults

The following parameters in `hdfs-site.xml` have new default values:

New Parameters

The following parameters have been added to `hdfs-site.xml`:

Name	Default Value
dfs.namenode.retrycache.heap.percent	0.03f
dfs.client.write.exclude.nodes.cache.expiry.interval.millis	600000
dfs.namenode.retrycache.expirytime.millis	600000
dfs.image.transfer.timeout	600000
dfs.namenode.enable.retrycache	true
dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction	0.75f
dfs.namenode.edits.noeditlogchannelflush	false
dfs.namenode.fs-limits.max-blocks-per-file	1048576
dfs.namenode.fs-limits.min-block-size	1048576
dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold	10737418240

mapred-site.xml

Changed Defaults

The following parameters in `mapred-default.xml` have new default values:

Name	Old Default Value	New Default Value
mapreduce.shuffle.port	8080	13562
yarn.app.mapreduce.client-am.ipc.max-retries	1	3
mapreduce.application.classpath	$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*, $HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*	No default value

New Parameters

The following parameters have been added to `mapred-site.xml`:

Name	Default Value
mapreduce.jobhistory.loadedjobs.cache.size	5
mapreduce.am.max-attempts	2
mapreduce.jobhistory.done-dir	${yarn.app.mapreduce.am.staging-dir}/history/done
mapreduce.jobhistory.cleaner.enable	true
mapreduce.jobhistory.datestring.cache.size	200000
mapreduce.jobhistory.max-age-ms	604800000
mapreduce.job.token.tracking.ids.enabled	false
mapreduce.jobhistory.joblist.cache.size	20000
mapreduce.jobhistory.move.thread-count	3

Name	Default Value
mapreduce.jobhistory.cleaner.interval-ms	86400000
mapreduce.jobhistory.client.thread-count	10
mapreduce.jobhistory.move.interval-ms	180000
mapreduce.jobhistory.minicluster.fixed.ports	false
mapreduce.jobhistory.http.policy	HTTP_ONLY
mapreduce.jobhistory.intermediate-done-dir	${yarn.app.mapreduce.am.staging-dir}/history/done_intermediate

httpfs-site.xml

New Parameters

The following parameters have been added to `httpfs-site.xml`:

Name	Default Value
httpfs.user.provider.user.pattern	^[A-Za-z_][A-Za-z0-9._-]*[$]?$

capacity-scheduler.xml

Changed Defaults

The following parameters in `capacity-scheduler.xml` have new default values:

Name	Old Default Value	New Default Value
yarn.scheduler.capacity.resource-calculator	org.apache.hadoop.yarn.server.resourcemanager.resource.DefaultResourceCalculator	org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator

hbase-site.xml

Changed Defaults

The following parameters in `hbase-site.xml` have new default values:

Name	Old Default Value	New Default Value
hbase.client.pause	1000	100
hbase.client.retries.number	10	35
hbase.client.scanner.caching	1	100
hbase.hregion.majorcompaction	86400000	604800000
hbase.hstore.blockingStoreFiles	7	10
hbase.regionserver.checksum.verify	false	true
hbase.regionserver.global.memstore.lowerLimit	0.35	0.38
hbase.regionserver.handler.count	10	30
hbase.regionserver.hlog.reader.impl	org.apache.hadoop.hbase.regionserver.wal.SequenceFileLogReader	org.apache.hadoop.hbase.regionserver.wal.ProtobufLogReader

Name	Old Default Value	New Default Value
hbase.regionserver.hlog.writer.impl	org.apache.hadoop.hbase.regionserver.wal.SequenceFileLogWriter	org.apache.hadoop.hbase.regionserver.wal.Prot
hbase.rootdir	file:///tmp/hbase-${user.name}/hbase	${hbase.tmp.dir}/hbase
hfile.block.cache.size	0.25	0.4
zookeeper.session.timeout	180000	90000

New Names

The following parameters in `hbase-site.xml` have new names:

Old Name	New Name	Default Value
hbase.rpc.engine	hbase.rpc.server.engine	org.apache.hadoop.hbase.ipc.WritableRpcEngi -> org.apache.hadoop.hbase.ipc.ProtobufRpcServ
io.storefile.bloom.cacheonwrite	hfile.block.bloom.cacheonwrite	false (See *https://issues.apache.org/jira/browse/HBASE-5957*)

Removed Parameters

The following parameters have been removed from `hbase-site.xml`:

Name	Default Value	Description
hbase.table.archive.directory	.archive	Removed due to *https://issues.apache.org/jira/browse/HBASE-8195*
hbase.regionserver.separate.hlog.for.meta	false	
dfs.support.append	true	HDFS now support append by default.
hbase.mapreduce.hfileoutputformat.blocksize	65536	
hbase.regionserver.nbreservationblocks	4	
hbase.regionserver.lease.period	60000	
hbase.hash.type	murmur	
hbase.regionserver.class	org.apache.hadoop.hbase.ipc.HRegionInterface	

New Parameters

The following parameters have been added to `hbase-site.xml`:

Name	Default Value
hbase.client.scanner.timeout.period	60000
hbase.storescanner.parallel.seek.enable	false
hbase.thrift.htablepool.size.max	1000
hbase.hstore.bytes.per.checksum	16384

Name	Default Value
hbase.config.read.zookeeper.config	false
hbase.master.loadbalancer.class	org.apache.hadoop.hbase.master.balancer.StochasticLoadBalanc
hbase.rpc.shortoperation.timeout	10000
hbase.snapshot.enabled	true
hbase.hstore.checksum.algorithm	CRC32
hbase.status.publisher.class	org.apache.hadoop.hbase.master.ClusterStatusPublisher $MulticastPublisher
hbase.status.listener.class	org.apache.hadoop.hbase.client.ClusterStatusListener $MulticastListener
hbase.security.authentication	simple
hbase.master.catalog.timeout	600000
hbase.hstore.compaction.kv.max	10
fail.fast.expired.active.master	false
hbase.metrics.exposeOperationTimes	true
hbase.client.localityCheck.threadPoolSize	2
hbase.status.published	false
hbase.status.multicast.address.ip	226.1.1.3
hbase.dynamic.jars.dir	${hbase.rootdir}/lib
hbase.hregion.majorcompaction.jitter	0.50
hbase.status.multicast.address.port	6100
hbase.lease.recovery.dfs.timeout	64000
hbase.server.compactchecker.interval.multiplier	1000
hbase.rpc.timeout	60000
hbase.lease.recovery.timeout	900000
hbase.storescanner.parallel.seek.threads	10
hbase.regionserver.catalog.timeout	600000
hbase.ipc.client.tcpnodelay	true
hbase.rest.filter.classes	org.apache.hadoop.hbase.rest.filter.GzipFilter
hbase.ipc.client.fallback-to-simple-auth-allowed	false
hbase.table.lock.enable	true

hive-site.xml

New Parameters

The following parameters have been added to `hive-site.xml`:

Name	Default Value
hive.default.rcfile.serde	org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe

Chapter 9

Administering PHD Using the CLI

This section describes the administrative actions that can be performed via Pivotal Command Center's command line interface (CLI).

Managing a PHD Cluster

This section describes the tasks you can perform from the CLI to manage a PHD cluster.

Starting a Cluster

You can use the `icm_client start` command to:

- Start all the configured services of the cluster.
- Start individual services configured for the cluster.
- Start individual roles on a specific set of hosts.

The command starts all configured cluster services in the right topological order based on service dependencies.

Note: You cannot start GemFire XD (gfxd) using the `icm_client start` command.

See the *GemFire XD documentation* for information about how to configure and start GemFire XD members.

Syntax

```
icm_client start --help
Usage: /usr/bin/icm_client start [options]

Options:
  -h, --help            show this help message and exit
  -v, --verbose         increase output verbosity
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        the name of the cluster on which the operation is
                        performed
  -s SERVICES, --service=SERVICES
                        service to be started
  -f, --force           forcibly start cluster (even if install is
 incomplete)
  -r ROLES, --role=ROLES
                        The name of the role which needs to be started
  -o HOSTFILE, --hostfile=HOSTFILE
                        The absolute path for the file containing host names
                        for the role which needs to be started
```

Options

This section describes the `start` options for the HDFS, MapRed, ZooKeeper, HBase, and HAWQ services.

-s

Starts the specified service and all services it depends on in the right topological order. The supported services are HDFS, Yarn, Zookeeper, Hbase, Hive, HAWQ, Pig, and Mahout.

-r

Starts only the specified role on a specific set of hosts. Hosts can be specified using the –o option.

-f

Forces the cluster to start even if the installation is incomplete.

Notes

The first time the cluster is started, Pivotal HD implicitly initializes the cluster. For subsequent invocations of the `start` command, the cluster is not initialized.

Cluster initialization includes the following:

- NameNode format.
- Create directories on the local filesystem of cluster nodes and on the hdfs, with the correct permission overrides. See the *Overriding Directory Permissions* section.
- Create HDFS directories for additional services, such as HBase, if these are included in the configured services.

Note: Refer to the "Verifying the Cluster Nodes for Pivotal HD" section to make sure the cluster services are up and running.

Make sure you back up all the data prior to installing or starting a new cluster on nodes that have pre-existing data on the configured mount points.

Examples

Cluster level start:

```
[gpadmin]# icm_client start -l <CLUSTERNAME>
```

Service level start:

```
[gpadmin]# icm_client start -l <CLUSTERNAME> -s hdfs
```

Role level start:

```
[gpadmin]# icm_client start -l <CLUSTERNAME> -r datanode -o hostfile
```

Stopping a Cluster

You can use the `icm_client stop` command to stop an entire cluster, to stop a single service, and to stop a single role on a specific set of hosts on which it is configured.

The command stops all configured cluster services in the right topological order, based on service dependencies.

Syntax

```
[gpadmin]# icm_client stop -h
Usage: icm_client stop [options]

Options:
  -h, --help            Show this help message and exit
  -v, --verbose         Increase output verbosity
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        The name of the cluster on which the operation is
                        performed
  -s SERVICES, --service=SERVICES
                        Service to be stopped
  -r ROLES, --role=ROLES
                        The name of the role which needs to be stopped
  -o HOSTFILE, --hostfile=HOSTFILE
                        The absolute path for the file containing host names
                        for the role that needs to be stopped
```

Options

This section describes the `stop` options for the HDFS, MapRed, ZooKeeper, HBase, and HAWQ services.

-s

Stops the specified service and all services it depends on in the right topological order. The supported services are HDFS, Yarn, Zookeeper, Hbase, Hive, HAWQ, Pig, and Mahout.

-r

Stops only the specified role on a specific set of hosts. Hosts can be specified using the `-o` option.

Examples

Cluster level stop:

```
[gpadmin]# icm_client stop -l <CLUSTERNAME>
```

Service level stop:

```
[gpadmin]# icm_client stop -l <CLUSTERNAME> -s hdfs
```

Role level stop:

```
[gpadmin]# icm_client stop -l <CLUSTERNAME> -r datanode -o hostfile
```

Restarting a Cluster

You can use the `icm_client restart` command to stop, then restart, a cluster.

See *Starting a Cluster* on page 112 and *Stopping a Cluster* on page 113 for more details about the stop/start operations.

Syntax

```
[gpadmin]#  icm_client restart -h
Usage: /usr/bin/icm_client restart [options]

Options:
  -h, --help            Show this help message and exit
  -v, --verbose           Increase output verbosity
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                          The name of the cluster on which the operation is
                          performed
  -s SERVICES, --service=SERVICES
                          The service to be restarted
  -f, --force             Forcibly start cluster (even if install is
 incomplete)
  -r ROLES, --role=ROLES
                          The name of the role which needs to be started
  -o HOSTFILE, --hostfile=HOSTFILE
                          The absolute path for the file containing host names
                          for the role which needs to be started
```

Reconfiguring a Cluster

Run the `icm_client reconfigure` command to update specific configurations for an existing cluster.

Some cluster-specific configurations cannot be updated:

Note:

- Reconfiguring the topology of a cluster (host-to-role mapping) is not allowed. For example, you cannot change the NameNode to a different node or add a new set of datanodes to a cluster.
- Properties based on hostnames: For example, `fs.defaultFS`, `dfs.namenode`. and the `http-address`.
- Properties with directory paths as values.

The following table lists properties that can only be changed with the `-f | --force` option:

Property Name	Configuration File
datanode.disk.mount.points	clusterConfig.xml
namenode.disk.mount.points	clusterConfig.xml
secondary.namenode.disk.mount.points	clusterConfig.xml
hawq.master.directory	clusterConfig.xml
hawq.segment.directory	clusterConfig.xml
zookeeper.data.dir	clusterConfig.xml

Note:

- You are expected to take care of all the necessary prerequisites prior to making changes to any of the following properties by using the `force` option.
- Incorrect provisioning can put the cluster into an inconsistent/unusable state.

Syntax

```
[gpadmin]#  icm_client reconfigure -h
Usage: /usr/bin/icm_client reconfigure [options]

Options:
  -h, --help              show this help message and exit
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                          the name of the cluster on which the operation is
                          performed
  -c CONFDIR, --confdir=CONFDIR
                          Directory path where cluster configuration is stored
  -s, --noscanhosts       Do not verify cluster nodes.
```

```
 -p, --nopreparehosts  Do not preparehosts as part of deploying the
cluster.
 -j JDKPATH, --java=JDKPATH
                       Location of Sun Java JDK RPM (Ex: jdk-
                       7u15-linux-x64.rpm). Ignored if -p is specified
 -t, --ntp             Synchronize system clocks using NTP. Optionally
takes
                       NTP server as argument. Defaults to pool.ntp.org
                       (requires external network access). Ignored if -p is
                       specified
 -d, --selinuxoff      Disable SELinux. Ignored if -p is specified
 -i, --iptablesoff     Disable iptables. Ignored if -p is specified
 -y SYSCONFIGDIR, --sysconf=SYSCONFIGDIR
                       [Only if HAWQ is part of the deploy] Directory
                       location of the custom conf files (sysctl.conf and
                       limits.conf) which will be appended to
                       /etc/sysctl.conf and /etc/limits.conf on slave
nodes.
                       Default: /usr/lib/gphd/gphdmgr/hawq_sys_config/.
                       Ignored if -p is specified
 -f, --force           Forcibly reconfigure the cluster (allows changes to
                       any servicesConfigGlobals property)
```

Reconfiguring an Existing Cluster

To reconfigure an existing cluster:

1. Stop the cluster:

   ```
   icm_client stop -l <CLUSTERNAME>
   ```

2. Fetch the configurations for the cluster into a local directory:

   ```
   icm_client fetch-configuration -l <CLUSTERNAME> -o <LOCALDIR>
   ```

3. Edit the configuration files in the cluster configuration directory (`<LOCALDIR>`).
4. Reconfigure the cluster:

   ```
   icm_client reconfigure -l <CLUSTERNAME> -c <LOCALDIR>
   ```

Synchronizing Configuration Files

Following an upgrade or reconfiguration, you need to synchronize the configuration files:

1. Fetch the new templates that come with the upgraded software by running:

   ```
   icm_client fetch-template
   ```

2. Retrieve the existing configuration from the database using:

   ```
   icm_client fetch-configuration
   ```

3. Synchronize the new configurations (`hdfs/hadoop-env`) from the template directory to the existing cluster configuration directory.
4. Upgrade or reconfigure the service by specifying the cluster configuration directory with updated contents.

Adding/Removing Services

Services can be added/removed using the `icm_client reconfigure` command.

- Edit the `clusterConfig.xml` file to add or remove services from the service list in the `services` tag.
- Edit the `hostRoleMapping` section to add or remove hosts for the specific services configured.
- Edit the `servicesConfigGlobals` if required for the specific service added.
- Follow the steps for *Reconfiguring a Cluster* on page 115.
- In a new deployment, you can use the `-p` or `-s` option to disable scanhosts or preparehosts on the newly added hosts.
- If you want to prepare the new hosts with Java, or if you want to disable iptables or SELinux, follow the instructions for installing Java mentioned in the *Deploying a Cluster* section of this document.

Note: Removing a specific service using the `icm_client reconfigure` command does not remove RPMs from the nodes. The RPMs are only removed when the cluster is uninstalled

Adding Hosts to a Cluster

If you plan to add hosts as part of adding a new service, perform the following tasks:

- Prepare the new hosts using the `icm_client preparehosts` command.
- Refer to *Adding/Removing Services* on page 116.

If you plan to add/remove hosts, as part of an existing service in the cluster, do the following:

Note: You can only add or remove hosts for slave roles (refer to *Expanding a Cluster* on page 118 for the list of slave roles). You cannot make host changes for any other role.

- Prepare the new hosts using the `icm_client preparehosts` command.
- You can add the new hosts to the corresponding slave roles in the `hostRoleMapping` section in `clusterConfig.xml`.
- Follow the steps for *Reconfiguring a Cluster* on page 115.

Note: You cannot add one service and remove another at the same time. You have to perform these as two separate steps; however, you can add multiple services OR remove multiple services at the same time.

Retrieving Information about a Deployed Cluster

Run the `icm_client fetch-configuration` command to fetch the configurations for an existing cluster and store them in a local file system directory.

Syntax

```
icm_client fetch-configuration -h
Usage: icm_client fetch-configuration [options]

Options:
  -h, --help            show this help message and exit
  -o OUTDIR, --outdir=OUTDIR
                        Directory path to store the cluster configuration
                        template files
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        Name of the deployed cluster whose configurations
 need
                        to be fetched
```

Examples

```
icm_client fetch-configuration -l <CLUSTERNAME> -o <LOCALDIR>
```

Listing Clusters

Run the `icm_client list` command to see a list of all the installed clusters.

Syntax

```
[gpadmin]# icm_client list --help
Usage: icm_client list [options]

Options:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
```

Examples

```
icm_client list
```

Expanding a Cluster

Note:

- Make sure you run `icm_client preparehosts` against the new slave hosts prior to adding them to the cluster. (See the `icm_client preparehosts` command example in the "Preparing the Cluster for Pivotal HD" section.)
- If security is enabled on the cluster; you will have to re-enable it after adding a node.

Run the `icm_client add-slaves` command to add additional slave hosts to an existing cluster. All the slave roles for *existing* cluster services will be installed on the new cluster hosts.

The following table indicates the services and their corresponding slave roles. Services not included in this list are not allowed for expansion (or shrinking).

Service Name	Slave
hdfs	datanode
yarn	yarn-nodemanager
hbase	hbase-regionserver
hawq	hawq-segment

If you only want to install an individual component on a node, you should do this by manually editing the `clusterConfig.xml` file, then running the `icm_client reconfigure` command (see *Reconfiguring a Cluster* on page 115).

Syntax

```
icm_client add-slaves --help
Usage: /usr/bin/icm_client add-slaves [options]

Options:
  -h, --help            show this help message and exit
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        the name of the cluster on which the operation is
                        performed
  -f HOSTFILE, --hostfile=HOSTFILE
```

```
                        file containing new-line separated list of hosts
 that
                        are going to be added.
  -s, --noscanhosts     Do not verify cluster nodes.
  -j JAVAHOME, --java_home=JAVAHOME
                        JAVA_HOME path to verify on cluster nodes
  -p, --nopreparehosts  Do not preparehosts as part of deploying the
 cluster.
  -k JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM (Ex: jdk-
                        7u15-linux-x64.rpm). Ignored if -p is specified
  -t, --ntp             Synchronize system clocks using NTP. Optionally
 takes
                        NTP server as argument. Defaults to pool.ntp.org
                        (requires external network access). Ignored if -p is
                        specified
  -d, --selinuxoff      Disable SELinux for the newly added nodes. Ignored
 if
                        -p is specified
  -i, --iptablesoff     Disable iptables for the newly added nodes. Ignored
 if
                        -p is specified
  -y SYSCONFIGDIR, --sysconf=SYSCONFIGDIR
                        [Only if HAWQ is part of the deploy] Directory
                        location of the custom conf files (sysctl.conf and
                        limits.conf) which will be appended to
                        /etc/sysctl.conf and /etc/limits.conf of the newly
                        added slave nodes. Default:
                        /usr/lib/gphd/gphdmgr/hawq_sys_config/. Ignored if -
p
                        is specified
```

Examples

```
icm_client add-slaves -l <CLUSTERNAME> -f slave_hostfile
```

After adding slave hosts, make sure you start the DataNode and Yarn nodemanager on the newly added slave hosts.

```
icm_client start -l <CLUSTERNAME> -r datanode -o hostfile
icm_client start -l <CLUSTERNAME> -r yarn-nodemanager -o hostfile
```

Important:

- If HBase is configured, start `hbase-regionservers` as well.
- Don't expect data blocks to be distributed to the newly added slave nodes immediately.

Note: If HAWQ is configured, see *Expanding HAWQ* on page 145.

Note: Hive does not have any slave roles, and therefore cannot be provisioned for an expansion.

Shrinking a Cluster

Note: Make sure you decommission the slave hosts (see *Decommissioning Slave Nodes* on page 120) prior to removing them, to avoid potential data loss.

Shrink a cluster by running the `icm_client remove-slaves` command, which removes slave hosts from an existing cluster. All the slave roles for the existing cluster services will be removed from the given hosts.

Syntax

```
icm_client remove-slaves --help
Usage: /usr/bin/icm_client remove-slaves [options]

Options:
  -h, --help            show this help message and exit
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        the name of the cluster on which the operation is
                        performed
  -f HOSTFILE, --hostfile=HOSTFILE
                        file containing new-line separated list of hosts
 that
                        are going to be removed.
```

Examples

```
icm_client remove-slaves -l <CLUSTERNAME> -f hostfile
```

Decommissioning Slave Nodes

Decommissioning is required to prevent potential loss of data blocks when you shutdown/remove slave hosts from a cluster.

Decommission Nodes Overview

The Hadoop distributed scale-out cluster-computing framework was inherently designed to run on commodity hardware with typical JBOD configuration (just a bunch of disks; a disk configuration where individual disks are accessed directly by the operating system without the need for RAID). The idea behind it relates not only to cost, but also fault-tolerance where nodes (machines) or disks are expected to fail occasionally without bringing the cluster down. Because of these reasons, Hadoop administrators are often tasked to decommission, repair, or even replace nodes in a Hadoop cluster.

Decommissioning slave nodes is a process that is used to prevent data loss when you need to shutdown or remove these nodes from a Pivotal HD Cluster. For instance, if multiple nodes need to be taken down, there is a possibility that all the replicas of one or more data blocks live on those nodes. If the nodes are just taken down without preparation, those blocks will no longer be available to the active nodes in the cluster, and so the files that contain those blocks will be marked as corrupt and will appear as unavailable.

Hadoop Administrators may also want to decommission nodes to shrink an existing cluster or proactively remove nodes. The process of decommission is not an instant process since it will require the replication of all of the blocks on the decommissioned node(s) to active nodes that will remain in the cluster. Decommissioning nodes should only be used in cases where more than one node needs to be taken down for maintenance, because it evacuates the blocks from the targeted hosts and can affect both data balance and data locality for Hadoop and higher level services, such as HAWQ (see instructions and recommendations specific to HAWQ in the following topics).

Related Tasks:

Decommissioning the Data Node on page 121

Decommissioning the YARN NodeManager on page 121

Shutting Down the Slave Node on page 122

Decommissioning the Data Node

These procedures assume that Name Node High Availability (HA) is enabled (a Pivotal best practice, and in PHD 2.1 and higher, the default configuration). If HA is not enabled, skip the additional steps for the Standby Name Node.

We recommend that you run a filesystem check on HDFS to verify the filesystem is healthy before you proceed with decommissioning any nodes. As `gpadmin`, run:

```
gpadmin# sudo -u hdfs hdfs fsck /
```

On the Active Name Node:

1. Edit the `/etc/gphd/hadoop/conf/dfs.exclude` file and add the Data Node hostnames to be removed (separated by a newline character). Make sure you use the fully qualified domain name (FQDN) for each hostname.
2. Instruct the Active NameNode to refresh its nodelist by re-reading the `.exclude` and `.include` files:

```
gpadmin# sudo -u hdfs hdfs dfsadmin –fs hdfs://<active_namenode_fqdn> –
refreshNodes
```

On the Standby NameNode:

1. Edit the `/etc/gphd/hadoop/conf/dfs.exclude` file and add the DataNode hostnames to be removed (separated by a newline character). Make sure you use the FQDN for each hostname.
2. Instruct the Standby NameNode to refresh its nodelist by re-reading the `.exclude` and `.include` files:

```
gpadmin# sudo -u hdfs hdfs dfsadmin -fs hdfs://<standby_namenode_fqdn> –
refreshNodes
```

Check the Decommission Status:

You can monitor the decommission progress by accessing the Name Node WebUI (`http://<active_namenode_host>:50070`) and navigating to the **Decommissioning Nodes** page.

You can also monitor the status via the command line by executing one of the following commands on any Name Node or Data Node in the cluster (verbose/concise):

```
gpadmin# sudo -u hdfs hdfs dfsadmin –report
gpadmin# sudo -u hdfs hdfs dfsadmin -report | grep -B 2 Decommission
```

Check whether the admin state has changed to **Decommission in Progress** for the Data Nodes being decommissioned. When all the Data Nodes report their state as **Decommissioned**, then all the blocks have been replicated.

Next step:

Decommissioning the YARN NodeManager on page 121

Decommissioning the YARN NodeManager

Previous Step:

Decommissioning the Data Node on page 121

Use the following procedure if YARN NodeManager daemons are running on the nodes that are being decommissioned.

Note that this process is almost immediate and only requires a notification to the ResourceManager that the excluded nodes are no longer available for use.

On the Yarn ResourceManager host machine:

1. Edit the `/etc/gphd/hadoop/conf/yarn.exclude` file and add the node manager hostnames to be removed (separated by newline character). Make sure you use the FQDN for each hostname.
2. On the Resource Manager host instruct the Resource Manager to refresh its node list by re-reading the `.exclude` and `.include` files:

   ```
   gpadmin# sudo -u yarn yarn rmadmin -refreshNodes
   ```

Check the Decommission Status:

You can verify the decommission state via the Resource Manager WebUI (`https://<resource_manager_host>:8088`) or by using the command line by executing the following command on the Resource Manager host:

```
gpadmin# sudo -u yarn yarn rmadmin node -list
```

Next Step:

Shutting Down the Slave Node on page 122

Shutting Down the Slave Node

Previous Step:

Decommissioning the YARN NodeManager on page 121

Once the slave nodes have been decommissioned, the slave processes running on the newly decommissioned nodes need to be shutdown via the Pivotal Command Center CLI.

To shut down the slave node:

- *Create a Hostfile*
- *Shut Down the Processes*
  - *If the hosts are HDFS DataNodes*
  - *If the hosts are YARN NodeManagers*
  - *If the hosts are HBase RegionServers*
  - *If the hosts are GemfireXD Servers*
  - *If the hosts are HAWQ Segment Servers*
    - *Data in Place*
    - *Data Removed*

Create a Hostfile

Create a text file containing the hostnames that have been decommissioned (separated by a newline character).

Make sure you use the FQDN for each hostname (`hostfile.txt`):

Shut Down the Processes

Shutdown the processes on the decommissioned nodes as follows:

If the hosts are HDFS DataNodes

Run:

```
gpadmin# icm_client stop -r datanode -r datanode -o <hostfile.txt>
```

If the hosts are YARN NodeManagers

Run:

```
gpadmin# icm_client stop -l <CLUSTERNAME> -r yarn-nodemanager -o
 <hostfile.txt>
```

If the hosts are HBase RegionServers

It is preferable to use the `graceful_stop.sh` script that HBase provides. The script checks to see if the Region Load Balancer is operational turns it off before starting its region server decommission process. If you want to decommission more than one node at a time by stopping multiple RegionServers concurrently, the RegionServers can be put into a "draining" state to avoid offloading data to other servers being drained. This is done by marking a RegionServer as a draining node by creating an entry in ZooKeeper under the `<hbase_root>/draining` znode. This znode has the format `name,port,startcode`, like the regionserver entries under `<hbase_root>/rs` node.

Using zkCLI, list the current HBase Region Servers:

```
[zk:] ls /hbase/rs
```

Use the following command to put any servers you wish into draining status. Copy the entry exactly as it exists in the `/hbase/rs` znode:

```
[zk:] create /hbase/draining/<FQDN_Hostname>,<Port>,<startcode>
```

This process will ensure that these nodes don’t receive new blocks as other nodes are decommissioned.

If the hosts are GemfireXD Servers

Run:

```
gpadmin# gfxd server stop -dir=<working_dir_containing_status_file>
```

If the hosts are HAWQ Segment Servers

If HAWQ is deployed on the hosts, you need to consider data locality concerns before leveraging the HDFS DataNode decommission process. HAWQ leverages a hash distribution policy to distribute its data evenly across the cluster, but this distribution is negatively effected when the data blocks are evacuated to the other hosts throughout the cluster. If the DataNode is later brought back online, two states are possible:

Data in Place

In this case, when the DataNode is brought back online HDFS reports the blocks stored on the node as "over-replicated" blocks. HDFS will, over-time, randomly remove a replica of each of the blocks. This process may negatively impact the data locality awareness of the HAWQ segments, because data that hashes to this node could now be stored elsewhere in the cluster. Operations can resume in this state with the only impact being potential HDFS network reads for some of the data blocks that had their primary replica moved off the host as the "over-replication" is resolved. This will not, however, affect co-located database joins, because the segment servers will be unaware that the data is being retrieved via the network rather than a local disk read.

Data Removed

In this case, when the DataNode is brought back online HDFS will now use this node for net-new storage activities, but the pre-existing blocks will not be moved back into their original location. This process will negatively impact the data locality for the co-located HAWQ segments because any existing data will not be local to the segment host. This will not result in a database gather motion since the data will still appear to be local to the segment servers, but it will require the data blocks to be fetched over the network during the HDFS reads. HDFS Balancer should not be used to repopulate data onto the newly decommissioned server unless a HAWQ table redistribution is planned as well. The HDFS Balancer will affect segment host data locality on every node in the cluster as it moves data around to bring HDFS utilization in balance across the cluster.

In either case, a HAWQ table redistribution can be performed on specific tables, or all tables in order to restore data locality. If possible, it is recommended that maintenance on a cluster containing HAWQ should be done one host at a time to avoid the situations described above. This alleviates the need to decommission the host, because two valid replicas of the data would exist at all times.

There is no specific decommission process for a HAWQ segment host, but if the host needs to be decommissioned, the HAWQ segment servers should be shutdown.

- On the Decommissioned Node, stop the `postgres` processes and then verify they are down:

```
gpadmin# pkill -SIGTERM postgres
gpadmin# ps –ef | grep postgres
```

- On the HAWQ Master, verify the segments are down:

```
gpadmin# source /usr/local/hawq/greenplum_path.sh
gpadmin# gpstate
```

Replacing the Slave Node

There are many situations in which a slave node goes down and the entire server must be replaced. In these cases, the administrator is not able to issue a decommission, so HDFS will mark the server offline and begin replicating the now missing blocks to bring up replica count back within policy guidelines. To replace the node, a new server can be brought online with the same configuration (disk mounts, etc.) and the following procedure can be used on the PCC/ICM server to bring the replacement node into the cluster.

1. Get the current cluster configuration:

```
gpadmin# icm_client fetch-configuration -o <config_dir_target> -
l <CLUSTERNAME>
```

2. Remove the failed node from the cluster by creating a text file containing the fully qualified hostname of the host to replace and then running the ICM command below. This step is required even if the replacement node will have the same name, because adding a “net-new” node to the cluster will allow us to leverage the ICM automation to properly configure the replaced host.

```
gpadmin# icm_client remove-slaves -f <replaced_hostfile>.txt -
l <CLUSTERNAME>
```

3. Add the replaced host back into the cluster by using the original configuration from the first step:

```
gpadmin# icm_client add-slaves -f <replaced_hostfile>.txt -
l <CLUSTERNAME>
```

4. Manually start the slave processes on the newly replaced node:
   - If the node is a Data Node:

   ```
   gpadmin# icm_client start -r datanode -o <hostfile>.txt
   ```

   - If the node is a NodeManager:

   ```
   gpadmin# icm_client start -r yarn-nodemanager -o <hostfile>.txt
   ```

- If the node is a HBase Region Server:

  ```
  gpadmin# icm_client start -r hbase-regionserver -o <hostfile>.txt
  ```

- If the node is a HAWQ Segment Server:

  ```
  gpadmin# sudo massh <replaced_hostfile>.txt verbose “service hawq start"
  ```

  With HAWQ, the database engine needs to be informed that it now has the new segment server online, so you need to log in to the HAWQ Master and issue the appropriate recovery commands for HAWQ segments.

  On the HAWQ master:

  ```
  gpadmin# source /usr/local.hawq/greenplum_path.sh
  gpadmin# gprecoverseg -F -d <master_data_directory>
  ```

  These commands will bring the server back online, but refer to *If the hosts are HAWQ Segment Servers* on page 123 for how to proceed in regards to the data within the database instance itself.

Replacing the Slave Node Disk

Hadoop is extremely resilient in terms of hardware failure, but disk failure is one type of failure scenario that relies on the administrator to put some thought into as the system is configured. In the default configuration, Hadoop will blacklist the slave node if a single disk fails. In most cases, this response is an extreme reaction to a relatively inconsequential failure that is relatively common in large Hadoop clusters. The parameter to control this response is `dfs.datanode.failed.volumes.tolerated` and can be found in the `hdfs-site.xml` file. The value given to this parameter represents the number of HDFS DataNode directories can fail before the node is blacklisted. A good rule of thumb for this setting would be to tolerate 1 disk failure for every 6 data disks you have in the system. For example, a 12 disk server would have `dfs.datanode.failed.volumes.tolerated = 2`.

In the majority of scenarios with the proper failure tolerance configured, the disk will fail, but the DataNode will remain operational.

To replace the disk drive:

1. Stop DataNode, NodeManager, GemfireXD, and/or HAWQ processes using methods described in *Shutting Down the Slave Node* on page 122.
2. Replace the failed disk drive(s).
3. Follow the Slave Node Replacement procedures (`add-slaves`/`remove-slaves`) described in *Replacing the Slave Node* on page 124.

High Availability

This section describes how to disable, and re-enable High Availability (HA) on a cluster. This section also includes some best practices and command reference information for the `haadmin` command.

- Starting with PHD 2.1, HA is enabled by default for new installations.
- For upgrades, HA status is maintained between versions. If you upgrade from PHD 2.0.x where HA was disabled, the upgraded system will also have HA disabled.
- Currently, only Quorum Journal-based storage is supported for HA.
- Pivotal Command Center (PCC) 2.1 was the first version to support default HA. If you are running an earlier version, download and import the latest version of PCC. See *Installing PHD Using the CLI* on page 28 for details.
- HDFS commands need a Kerberos ticket when running in secure mode. See *Enabling Secure Mode Commands* on page 141 for more details.

Best Practices for High Availability

Before you deploy an HA cluster, you should take the following best practices into consideration:

- **NameNode machines:** The machines on which you run the Active and Standby NameNodes should have equivalent hardware to each other.
- **JournalNode machines:** The machines on which you run the JournalNodes. The JournalNode daemons should be co-located on machines with other Hadoop master daemons; for example NameNodes, YARN ResourceManager.

There must be at least three JournalNode (JN) daemons, since edit log modifications are written to a majority of JNs. This allows the system to tolerate the failure of a single machine. You may also run more than three JournalNodes, but in order to increase the number of failures the system can tolerate, you should run an odd number (3, 5, 7, etc.).

When running with *N* JournalNodes, the system can tolerate at most `(N - 1) / 2` failures and continue to function normally.

Note: In an HA cluster, the Standby NameNode also performs checkpoints of the namespace state; therefore, it is not necessary to configure a Secondary NameNode, CheckpointNode, or BackupNode in an HA cluster.

One benefit of this is that since a Secondary NameNode is not needed in an HA cluster; if you are reconfiguring a non-HA-enabled HDFS cluster to be HA-enabled you can reuse the hardware you had previously dedicated to the Secondary NameNode.

Disabling High Availability

Starting with PHD 2.1, High Availability is enabled by default for new installations.

Note: HDFS commands need a Kerberos ticket when running in secure mode. See *Enabling Secure Mode Commands* on page 141 for more details.

To disable High Availability:

1. Synchronize the active and standby NameNode data.

   On the NameNode, run:

   ```
   sudo -u hdfs hdfs dfsadmin -safemode enter
   sudo -u hdfs hdfs dfsadmin -saveNamespace
   ```
2. Stop the cluster.

   On the Admin node, run:

   ```
   icm_client stop -l <CLUSTERNAME>
   ```
3. For HAWQ users, stop HAWQ.

   From the HAWQ master, as `gpadmin`, run the following:

   ```
   /etc/init.d/hawq stop
   ```
4. Back up the NameNode data.

   On both the active and standby NameNode, copy `{dfs.namenode.name.dir}/current` to a backup directory.
5. Fetch the configurations for the cluster in a local directory:

   ```
   icm_client fetch-configuration -l <CLUSTERNAME> -o <LOCALDIR>
   ```
6. Edit `clusterConfig.xml` as follows:

   a. Uncomment out the `secondarynamenode` role in the `hdfs` service.
   b. Comment the `standbynamenode` and `journalnode` roles in the `hdfs` service.
   c. Uncomment or add the `secondary.namenode.disk.mount.points`.
   d. Comment the `nameservices`, `namenode1id`, `namenode2id`, `journalpath`, and `journalport` entries in `serviceConfigGlobals`.

7. Edit `hdfs/hdfs-site.xml` as follows:

   a. Comment the following properties:

```
<property>
  <name>dfs.nameservices</name>
  <value>${nameservices}</value>
</property>

<property>
  <name>dfs.ha.namenodes.${nameservices}</name>
  <value>${namenode1id},${namenode2id}</value>
</property>

<property>
  <name>dfs.namenode.rpc-address.${nameservices}.${namenode1id}</name>
  <value>${namenode}:8020</value>
</property>

<property>
  <name>dfs.namenode.rpc-address.${nameservices}.${namenode2id}</name>
  <value>${standbynamenode}:8020</value>
</property>

<property>
  <name>dfs.namenode.http-address.${nameservices}.${namenode1id}</name>
  <value>${namenode}:50070</value>
</property>

<property>
  <name>dfs.namenode.http-address.${nameservices}.${namenode2id}</name>
  <value>${standbynamenode}:50070</value>
</property>

<property>
  <name>dfs.namenode.shared.edits.dir</name>
  <value>qjournal://${journalnode}/${nameservices}</value>
</property>

<property>
  <name>dfs.client.failover.proxy.provider.${nameservices}</name>

 <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</
value>
</property>

<property>
  <name>dfs.ha.fencing.methods</name>
  <value>
  sshfence
  shell(/bin/true)
  </value>
</property>

<property>
  <name>dfs.ha.fencing.ssh.private-key-files</name>
  <value>/home/hdfs/.ssh/id_rsa</value>
```

```
</property>

<property>
  <name>dfs.journalnode.edits.dir</name>
  <value>${journalpath}</value>
</property>

<!-- Namenode Auto HA related properties -->
<property>
   <name>dfs.ha.automatic-failover.enabled</name>
   <value>true</value>
 </property>
<!-- END Namenode Auto HA related properties -->
```

b. Uncomment or add the following properties:

```
<property>
  <name>dfs.namenode.secondary.http-address</name>
  <value>${secondarynamenode}:50090</value>
  <description>
    The secondary namenode http server address and port.
  </description>
</property>
```

8. Edit `yarn/yarn-site.xml` as follows:

a. Comment the following property:

```
<property>
    <name>mapreduce.job.hdfs-servers</name>
    <value>hdfs://${nameservices}</value>
</property>
```

b. Add the following property:

```
<property>
     <name>mapreduce.job.hdfs-servers</name>
    <value>hdfs://${namenode}:${dfs.port}</value>
</property>
```

9. Edit `hdfs/core-site.xml` as follows:

a. Set the following property key value:

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://${namenode}:${dfs.port}</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

b. Comment the following property:

```
<property>
   <name>ha.zookeeper.quorum</name>
   <value>${zookeeper-server}:${zookeeper.client.port}</value>
 </property>
```

10. In `hbase/hbase-site.xml`, set the following property key value:

```
<property>
```

```
    <name>hbase.rootdir</name>
    <value>hdfs://${namenode}:${dfs.port}/apps/hbase/data</value>
    <description>The directory shared by region servers and into
    which HBase persists.  The URL should be 'fully-qualified'
    to include the filesystem scheme.  For example, to specify the
    HDFS directory '/hbase' where the HDFS instance's namenode is
    running at namenode.example.org on port 9000, set this value to:
    hdfs://namenode.example.org:9000/hbase.  By default HBase writes
    into /tmp.  Change this configuration else all data will be lost
    on machine restart.
    </description>
</property>
```

11. To disable HA for HAWQ, uncomment the default `DFS_URL` property and comment out `DFS_URL` in `hawq/gpinitsystem_config` as follows:

```
DFS_URL=${namenode}:${dfs.port}/hawq_data
#### For Non-HA comment the following line
#DFS_URL=${nameservices}/hawq_data
```

12. Comment the following properties in `hawq/hdfs-client.xml`:

```
 <property>
  <name>dfs.nameservices</name>
  <value>${nameservices}</value>
 </property>

 <property>
  <name>dfs.ha.namenodes.${nameservices}</name>
  <value>${namenode1id},${namenode2id}</value>
 </property>

 <property>
  <name>dfs.namenode.rpc-address.${nameservices}.${namenode1id}</name>
  <value>${namenode}:8020</value>
 </property>

 <property>
  <name>dfs.namenode.rpc-address.${nameservices}.${namenode2id}</name>
  <value>${standbynamenode}:8020</value>
 </property>
 <property>
  <name>dfs.namenode.http-address.${nameservices}.${namenode1id}</name>
  <value>${namenode}:50070</value>
 </property>

 <property>
  <name>dfs.namenode.http-address.${nameservices}.${namenode2id}</name>
  <value>${standbynamenode}:50070</value>
 </property>

 <property>
  <name>dfs.client.failover.proxy.provider.${nameservices}</name>

 <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</
value>
 </property>
```

13. Change owner and permissions for the `container-executor.cfg` file located in `/etc/gphd/hadoop/conf` on all cluster nodes, if the file is present:

```
chmod 644 /etc/gphd/conf.gphd-2.0.1/container-executor.cfg
chown root:root /etc/gphd/conf.gphd-2.0.1/container-executor.cfg
```

14. Run the following command to reconfigure the cluster with your new configuration file:

```
icm_client reconfigure -l <CLUSTERNAME> -c <LOCALDIR>
```

15. Start the cluster:

```
icm_client start -l <CLUSTERNAME>
```

16. Update the HIVE Metastore.

    Hive metastore contains references to the `hdfs` path with `nameservices` in the URL. This needs to be updated to use `namenode:port`.

    Note: Make sure metastore is not running and is backed up to a persistent store before running the update commands.

    a. Log in to the host configured as hive-metastore.

    b. Display the current NameNode and hdfspath for the Hive warehouse directory:

    ```
    /usr/lib/gphd/hive/bin/metatool -listFSRoot
    ```

    c. Run the following command:

    ```
    /usr/lib/gphd/hive/bin/metatool -updateLocation
     hdfs://<current_namenode>:<dfs_port> hdfs://<nameservices>
    ```

    Where:

    <nameservices> is the logical name used for the nameservices in a HA-enabled cluster.

    <current_namenode> is the hostname of the NameNode on the cluster after reconfiguring to disable HA.

    Note: When specifying <*nameservices*>, do not use underscores ('_'); for example, `phd_cluster`.

17. For HAWQ users, restart HAWQ services for your configuration changes to take effect.

    From the HAWQ master, as `gpadmin`, run the following:

    ```
    /etc/init.d/hawq start
    ```

Related Topics:

Best Practices for High Availability on page 126

Enabling/Re-enabling High Availability on page 130

High Availability Command Reference on page 135

Enabling/Re-enabling High Availability

Prerequisites

Before you enable HA for any cluster:

- Make sure you take into consideration our recommended *Best Practices for High Availability* on page 126.
- Checkpoint your NameNode:
    - Stop all incoming data traffic.

- With the namenode running and the secondaryname node stopped, force checkpoint by running the following on the secondarynamenode:

  ```
  sudo -u hdfs hdfs secondarynamenode -checkpoint force
  ```

Enabling High Availability

To re-enable HA on a cluster:

1. For HAWQ users, stop HAWQ.From the HAWQ master, as `gpadmin`, run the following:

   ```
   /etc/init.d/hawq stop
   ```

2. Stop the cluster:

   ```
   icm_client stop -l <CLUSTERNAME>
   ```

3. Back up the NameNode data. Copy `{dfs.namenode.name.dir}/current` to a backup directory.
4. Fetch the configurations for the cluster in a local directory:

   ```
   icm_client fetch-configuration -l <CLUSTERNAME> -o <LOCALDIR>
   ```

5. Edit `clusterConfig.xml` as follows:
   a. Comment out the `secondarynamenode` role in the `hdfs` service.
   b. Uncomment the `standbynamenode` and `journalnode` roles in the `hdfs` service.
   c. Uncomment the `nameservices`, `namenode1id`, `namenode2id`, `journalpath`, and `journalport` entries in `serviceConfigGlobals`.
6. Edit `hdfs/hdfs-site.xml` as follows:

   Note: These edits are are for enabling automatic HA. If you want to enable manual HA, keep the Namenode Auto HA related properties commented out.

   a. Uncomment the following properties:

   ```
   <property>
     <name>dfs.nameservices</name>
     <value>${nameservices}</value>
   </property>

   <property>
     <name>dfs.ha.namenodes.${nameservices}</name>
     <value>${namenode1id},${namenode2id}</value>
   </property>

   <property>
     <name>dfs.namenode.rpc-address.${nameservices}.${namenode1id}</name>
     <value>${namenode}:8020</value>
   </property>

   <property>
     <name>dfs.namenode.rpc-address.${nameservices}.${namenode2id}</name>
     <value>${standbynamenode}:8020</value>
   </property>

   <property>
     <name>dfs.namenode.http-address.${nameservices}.${namenode1id}</name>
     <value>${namenode}:50070</value>
   </property>

   <property>
     <name>dfs.namenode.http-address.${nameservices}.${namenode2id}</name>
   ```

```
  <value>${standbynamenode}:50070</value>
</property>

<property>
  <name>dfs.namenode.shared.edits.dir</name>
  <value>qjournal://${journalnode}/${nameservices}</value>
</property>

<property>
  <name>dfs.client.failover.proxy.provider.${nameservices}</name>

 <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</
value>
</property>

<property>
  <name>dfs.ha.fencing.methods</name>
  <value>
  sshfence
  shell(/bin/true)
  </value>
</property>

<property>
  <name>dfs.ha.fencing.ssh.private-key-files</name>
  <value>/home/hdfs/.ssh/id_rsa</value>
</property>

<property>
  <name>dfs.journalnode.edits.dir</name>
  <value>${journalpath}</value>
</property>

<!-- Namenode Auto HA related properties -->
<property>
   <name>dfs.ha.automatic-failover.enabled</name>
   <value>true</value>
 </property>
<!-- END Namenode Auto HA related properties -->
```

b. Comment the following properties:

```
<property>
  <name>dfs.namenode.secondary.http-address</name>
  <value>${secondarynamenode}:50090</value>
  <description>
    The secondary namenode http server address and port.
  </description>
</property>
```

7. In `yarn/yarn-site.xml`, set the following property/value:

```
<property>
    <name>mapreduce.job.hdfs-servers</name>
    <value>hdfs://${nameservices}</value>
</property>
```

8. Edit `hdfs/core-site.xml` as follows:

a. Set the following property/value:

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://${nameservices}</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

b. Uncomment following property:

```
<property>
   <name>ha.zookeeper.quorum</name>
   <value>${zookeeper-server}:${zookeeper.client.port}</value>
 </property>
```

Note: The previous edits are for enabling automatic high availability. If you want to enable manual high availability, you need to additionally comment out the following property in `hdfs/core-site.xml`:

```
<!--
<property>
  <name>ha.zookeeper.quorum</name>
  <value>$
  {zookeeper-server}
  :$
  {zookeeper.client.port}
  </value>
</property>
-->
```

9. In `hbase/hbase-site.xml`, set the following property/value:

```
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://${nameservices}/apps/hbase/data</value>
    <description>The directory shared by region servers and into
    which HBase persists.  The URL should be 'fully-qualified'
    to include the filesystem scheme.  For example, to specify the
    HDFS directory '/hbase' where the HDFS instance's namenode is
    running at namenode.example.org on port 9000, set this value to:
    hdfs://namenode.example.org:9000/hbase.  By default HBase writes
    into /tmp.  Change this configuration else all data will be lost
    on machine restart.
    </description>
</property>
```

10. To enable HA for HAWQ, comment out the default `DFS_URL` property and uncomment `DFS_URL` in `hawq/gpinitsystem_config` as follows:

```
#DFS_URL=${namenode}:${dfs.port}/hawq_data
#### For HA uncomment the following line
DFS_URL=${nameservices}/hawq_data
```

11. Add the following properties to `hawq/hdfs-client.xml`:

```
 <property>
  <name>dfs.nameservices</name>
  <value>${nameservices}</value>
 </property>

 <property>
  <name>dfs.ha.namenodes.${nameservices}</name>
  <value>${namenode1id},${namenode2id}</value>
 </property>
 <property>
  <name>dfs.namenode.rpc-address.${nameservices}.${namenode1id}</name>
  <value>${namenode}:8020</value>
 </property>

 <property>
  <name>dfs.namenode.rpc-address.${nameservices}.${namenode2id}</name>
  <value>${standbynamenode}:8020</value>
 </property>
 <property>
  <name>dfs.namenode.http-address.${nameservices}.${namenode1id}</name>
  <value>${namenode}:50070</value>
 </property>

 <property>
  <name>dfs.namenode.http-address.${nameservices}.${namenode2id}</name>
  <value>${standbynamenode}:50070</value>
 </property>

 <property>
  <name>dfs.client.failover.proxy.provider.${nameservices}</name>

 <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</
value>
 </property>
```

12. On the Standby NameNode, move `{dfs.namenode.name.dir}/current` to a backup directory (or delete).
13. Reconfigure the cluster:

```
icm_client reconfigure -l <CLUSTERNAME> -c <LOCALDIR> -f
```

Caution: Running the `reconfigure` command on a secure cluster disables security in PHD-1.1.0.0 and PHD-1.1.1.0.

14. Start the cluster:

```
icm_client start -l <CLUSTERNAME>
```

15. Update the HIVE Metastore.

Hive metastore contains references to the `hdfs` path with `namenode:port` in the URL. This needs to be updated to use `nameservices`, so HIVE scripts can work whenever NameNode failure happens.

Note: Make sure metastore is not running and is backed up to a persistent store before running the update commands.

a. Log in to the host configured as hive-metastore.

b. Display the current NameNode and hdfspath for the Hive warehouse directory:

```
 /usr/lib/gphd/hive/bin/metatool -listFSRoot
```

c. Run the following command:

```
/usr/lib/gphd/hive/bin/metatool -updateLocation hdfs://<nameservices>
 hdfs://<current_namenode>:<dfs_port>
```

Where:

<nameservices> is the logical name used for the nameservices in a HA-enabled cluster.

<current_namenode> is the hostname of the NameNode on the cluster before reconfiguring to enable HA.

Note: When specifying *<nameservices>*, do not use underscores ('_'); for example, `phd_cluster`.

16. Restart HAWQ services for your configuration changes to take effect. From the HAWQ master, as `gpadmin`, run the following:

```
/etc/init.d/hawq start
```

Related Topics:

Best Practices for High Availability on page 126

Disabling High Availability on page 126

High Availability Command Reference on page 135

High Availability Command Reference

Note: HDFS commands need a Kerberos ticket when running in secure mode. See *Secure Mode Commands* for more details.

`hdfs haadmin` prints help for all subcommands and options. *<serviceid>* is the logical name configured for each NameNode, as `namenode1id` and `namenode2id`, in `clusterConfig.xml`.

- Check the state of a given NameNode:

```
hdfs haadmin -getServiceState <serviceid>
```

For example:

```
hdfs haadmin -getServiceState nn1
```

- Transition a given NameNode to standby:

```
hdfs haadmin -transitionToStandby <serviceid>
```

For example:

```
hdfs haadmin -transitionToStandby nn1
```

- Transition a given NameNode to active:

```
hdfs haadmin -transitionToActive <serviceid>
```

For example:

```
hdfs haadmin -transitionToActive nn1
```

- Failover between two NameNodes:

```
hdfs haadmin -failover <serviceid> <serviceid>
```

For example:

```
hdfs haadmin -failover nn1 nn2
```

Security/Kerberos Authentication

This section describes how to enable/disable Kerberos authentication for PHD clusters.

Kerberos is a network authentication protocol that provides strong authentication for client/server applications using secret-key cryptography.

You can set up a Kerberos server during PHD installation, or can connect to an existing Kerberos server. See *PHD Install 2 - Configure Kerberos and LDAP* on page 32 for details.

Note: In secure mode, the default Hive server is hive-server2:

- When you enable security, we switch to hive-server2.
- When you disable security, we switch back to hive-server.

Enabling Kerberos Authentication

To enable security on a deployed, but unsecured, cluster, you need to set up a Kerberos server, as follows. If you already have a Kerberos server set up, you do not need to run this command, but you need to make security-specific edits to the Cluster configuration file.

Configuring Kerberos:

1. Stop the cluster:

```
[gpadmin]# icm_client stop -l <CLUSTERNAME>
```

2. On the Admin node, as `gpadmin`, run:

```
$ icm_client security -i
```

3. You will be prompted for the following information:

```
Do you wish to configure Kerberos Server? (y/n) [Yes]? yes
```

   Enter `no` if you do not wish to use the built-in Kerberos server. The remaining instructions assume you chose to configure the built-in Kerberos server.

4. Choose a realm for your Kerberos server; usually this will be your domain name. For example:

```
Enter REALM for Kerberos (ex PIVOTAL.IO): PIVOTAL.IO
```

5. Choose a login and password for your Kerberos server. You will need these if you ever need to manage the Kerberos server directly via the command line tool (`kadmin`). We recommend using `gpadmin`:

```
Enter username for Kerberos Server ADMIN [admin]: gpadmin
Enter new password for Kerberos Server ADMIN:
Re-enter the new password for Kerberos Server Admin:
Enter new MASTER password for KDC:
Re-enter new MASTER password for KDC:
```

6. You are now prompted to set up the built-in LDAP server:

```
[WARNING] Attempt to re-configure previously configure LDAP server may
 result in data or functionality loss
Do you wish to configure LDAP Server? (y/n) [Yes]? yes
```

7. Select a suitable base domain name (DN); usually this will be your domain name. For example:

```
Enter Domain name for LDAP base DN (ex pivotal.io): pivotal.io
```

8. Choose a login and password for the LDAP administrator. You will need these to add new users into the system, and also it will be needed if you ever need to manage the built-in LDAP server directly. We recommend using `gpadmin`:

```
Enter username for LDAP Administrator [Manager]: gpadmin
Enter new password for LDAP administrator:
Re-enter new password for LDAP administrator:
```

9. The installer will now install and configure the built-in Kerberos and LDAP server, based on the information you provided:

```
[INFO] Attempting to configure KDC and/or LDAP. It may take few minutes...
[DONE] Security components initialized successfully
```

10. You now need to add security-specific parameters/values to the configuration file. You can use `icm_client reconfigure` for this purpose.

Note: Make sure it runs successfully on all nodes before proceeding further.

To use `icm_client reconfigure` to update the configuration file, perform the following tasks on the Admin node:

a. Fetch the current configuration in to a directory named `SecureConfiguration`:

```
[gpadmin]# icm_client fetch-configuration -o SecureConfiguration -l
 <CLUSTERNAME>
```

b. Open the cluster configuration file and set the security parameter to `true`:

```
<securityEnabled>true</securityEnabled>
```

c. Locate the following section in the Global Services Properties:

```
<servicesConfigGlobals>:
<!-- Security configurations -->
 <!-- provide security realm. e.g. EXAMPLE.COM -->
    <security.realm></security.realm>
    <!-- provide the path of kdc conf file -->
    <security.kdc.conf.location>/etc/krb5.conf</
security.kdc.conf.location>
```

You need to add a valid value to the `<security.realm>` parameter.The default value for the `<security.kdc.conf.location>` parameter is valid if you are using the Kerberos server set up during *PHD Install 2 - Configure Kerberos and LDAP* on page 32; if you are using an existing Kerberos server, you need to add a value for that location.

11. Run `reconfigure` to push your changes to the cluster nodes:

```
[gpadmin]# icm_client reconfigure -l <CLUSTERNAME> -c SecureConfiguration
 -f
```

12. Start the cluster:

```
[gpadmin]# icm_client start -l <CLUSTERNAME>
```

13. If HAWQ is configured:

a. Start HAWQ:

```
$ /etc/init.d/hawq start
```

b. Make sure you have a kerberos principal for `gpadmin`.

c. Locate the HAWQ data directory:

i. On the HAWQ master, open `/etc/gphd/hawq/conf/gpinitsystem_config`.

ii. Locate `DFS_URL` and obtain the directory after `nameservice` or `namenode`. By default the value of this is `hawq_data`. We will refer to it as <*HAWQ_DATA_DIR*> for the purpose of this document.

d. Create <*HAWQ_DATA_DIR*> on HDFS:

i. Start the cluster using `icm_client`.

ii. Make sure HDFS service is up and running.

iii. As `gpadmin`, on the namenode or client machine, run the following:

```
kinit
hadoop fs -mkdir /<HAWQ_DATA_DIR>
hadoop fs -chown -R postgres:gpadmin /<HAWQ_DATA_DIR>
hadoop fs -mkdir /user/gpadmin
hadoop fs -chown gpadmin:gpadmin /user/gpadmin
hadoop fs -chmod 777 /user/gpadmin
kdestroy
```

e. Specify that security is enabled by running the following:

```
source /usr/local/hawq/greenplum_path.sh
gpconfig -c enable_secure_filesystem -v "on"
gpconfig --masteronly -c krb_server_keyfile -v "'/path/to/keytab/file'"
```

Note: The single quotes `'` after and before the double quotes `"` in the keytab string above are required.

f. Restart HAWQ:

```
$ /etc/init.d/hawq restart
```

At this point, security should be enabled and you may run test commands to validate data is still accessible in secure mode.

Disabling Kerberos Authentication

To disable Kerberos authentication for a cluster:

1. Stop the cluster:

```
[gpadmin]# icm_client stop -l <CLUSTERNAME>
```

2. If you have HBase installed and HBase-to-Zookeeper communication is secured (true in most cases), complete the following tasks.

Tables created while HBase is secure have ACLs set on them that only allow SASL authenticated users to modify them. In order to operate in non-secure mode, you must do the following:

Note: You can skip these steps if you don't have HBase installed.

a. Start *just* the Zookeeper service:

```
[gpadmin]# icm_client start -l <CLUSTERNAME> -s zookeeper
```

b. On HBase master:

i. Run the Zookeeper CLI:

```
[gpadmin]# sudo -u hbase hbase zkcli
```

ii. Check if there are any regions in transition. Output `[]` means there are NO regions in transition at the moment and you don't need to set ACL on this sub znode:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 0] ls /hbase/region-in-transition
[]
```

If there are regions in transition, either wait for them to finish (start the cluster again) or set ACL to make them controllable by `world`. Do this for all the regions. For example, if you see a region such as `156781230`:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 1] setAcl /hbase/region-in-tranistion/156781230 world:anyone:cdrwa
```

iii. Check if there are unassigned regions. If there are any, set ACL to be controllable by `world`:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 2] ls /hbase/unassigned
[123456789]
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 3] setAcl /hbase/unassigned/123456789 world:anyone:cdrwa
```

iv. Do this for the `/hbase` znode and *all* the sub-znodes under `/hbase` where ACL is set to anything other than `world:anyone:cdrwa`; otherwise, they won't be readable while security is disabled.

Note: If you're only disabling security temporarily for upgrade, and intend to enable it again after upgrade, you may skip setting ACLs on znodes.

For example, for the `/hbase/table` sub-znodes:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 4] ls /hbase/table
[hbase:meta, hbase:namespace, testtable]
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 5] getAcl /hbase/table/hbase:meta
'world,'anyone
:cdrwa
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 6] getAcl /hbase/table/testtable
'world,'anyone
:r
'sasl,'hbase
:cdrwa
# Here is testtable is not world writable and has SASL enabled.
 If you want to use this table while in non-secure mode, do the
 following.
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 7] setAcl /hbase/table/testtable world:anyone:cdrwa

# Verify ACL has been set
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTE
 8] getAcl /hbase/table/testtable
'world,'anyone
:cdrwa
```

Note: Alternatively, you can also remove the `/hbase` znode or any of its sub-znodes, such as `/hbase/table`, as they will be re-created when the HBase service is restarted. Also, this should only be done if HBase-master and HBase-region server were shut down properly and there is no transient state yet to be synced back.

Use this option with *extreme* caution and only if you're having trouble starting HBase service. Careless use may cause data loss.

To remove a znode (e.g. `/hbase/table`), run the following:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CO
 9] rmr /hbase/table
```

v. Quit the Zookeeper CLI on the HBase master node. You can now disconnect from HBase master:

```
[zk:
 node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTI
 10] quit
```

c. Stop the Zookeeper service from the ICM Admin node:

```
[gpadmin]# icm_client stop -l test -s zookeeper
```

3. You now need to remove security-related changes from the configuration file. You can use `icm_client reconfigure` for this purpose.

Note: Make sure it runs successfully on all nodes before proceeding further.

To use `icm_client reconfigure` to update the configuration file, perform the following tasks on the ICM Admin node:

a. Fetch the current configuration in to a directory named `SecureConfiguration`:

```
[gpadmin]# icm_client fetch-configuration -o SecureConfiguration -
l <CLUSTERNAME>
```

b. Make the following modifications in the configuration file to disable security (note that this parameter is set to `true` by default):

```
<securityEnabled>false</securityEnabled>
```

c. Run `icm_client reconfigure` to push your changes to the cluster nodes:

```
[gpadmin]# icm_client reconfigure -l <CLUSTERNAME> -c
 SecureConfiguration
```

4. Remove security from any manually-installed service by following the reverse of the instructions to enable them.

5. Start the cluster.

```
[gpadmin]# icm_client start -l <CLUSTERNAME>
```

6. If HAWQ is configured:

a. Specify that security is *not* enabled by running the following:

```
source /usr/local/hawq/greenplum_path.sh
gpconfig --masteronly -c enable_secure_filesystem -v "off"
```

b. Restart HAWQ:

```
$ /etc/init.d/hawq restart
```

7. After disabling security on an HA cluster, you must delete all files from `nm-local-dir/usercache`.

At this point, security should be disabled and you may run test commands to validate data is still accessible in non-secure mode.

Enabling Secure Mode Commands

HDFS commands need a Kerberos ticket when running in secure mode.

Verifying/Obtaining/Removing a Kerberos Ticket

To check if you have a valid ticket, run `klist`:

```
[gpadmin@client ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_500
Default principal: gpadmin@PHDDEV.LOCAL
Valid starting     Expires            Service principal
09/08/14 23:54:42  09/09/14 23:54:42  krbtgt/PHDDEV.LOCAL@PHDDEV.LOCAL
 renew until 09/15/14 23:54:42
```

Make sure the ticket is valid. If there is no ticket present in cache or it has expired, then you need to obtain a new ticket.

Obtain a new ticket by running `kinit`.

The following example obtains a ticket for user `gpadmin` (pre-existing in the Kerberos database):

```
[gpadmin@client ~]$ kinit
Password for gpadmin@PHDDEV.LOCAL:                      # ENTER password
 here
```

In order to remove an existing kerberos ticket, use `kdestory`:

```
[gpadmin@client ~]$ kdestroy
```

Enabling Secure Commands for Non-dfs.cluster.administrator Users

In secure mode, certain commands, such as `hdfs haadmin` can only be run by users belonging to `dfs.cluster.administrators`. However the `gpadmin` user does not belong to `dfs.cluster.administrators` by default. If you attempt to run these commands as `gpadmin`, they will fail.

For example, running `hdfs haadmin -failover nn1 nn2` (in which `nn1` and `nn2` are your NameNodes) will fail with the following error:

```
[gpadmin@client ~]$ hdfs haadmin -failover nn1 nn2
Operation failed: Disallowed RPC access from gpadmin@PHDDEV.LOCAL
 (auth:KERBEROS) at 192.168.243.110. Not listed in
 dfs.cluster.administrators
 at
 org.apache.hadoop.hdfs.tools.DFSZKFailoverController.checkRpcAdminAccess(DFSZKFailoverC
 at
 org.apache.hadoop.ha.ZKFCRpcServer.gracefulFailover(ZKFCRpcServer.java:93)
 at
 org.apache.hadoop.ha.protocolPB.ZKFCProtocolServerSideTranslatorPB.gracefulFailover(ZKF
 at org.apache.hadoop.ha.proto.ZKFCProtocolProtos$ZKFCProtocolService
$2.callBlockingMethod(ZKFCProtocolProtos.java:1548)
 at org.apache.hadoop.ipc.ProtobufRpcEngine$Server
$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:585)
 at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:928)
 at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2048)
 at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:2044)
 at java.security.AccessController.doPrivileged(Native Method)
 at javax.security.auth.Subject.doAs(Subject.java:415)
 at
 org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1491)
 at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2042)
```

In order to run such commands, you can create or modify the `dfs.cluster.administrators` property in `hdfs-site.xml` and reconfigure/deploy.

Alternately, you need to run these commands as the `hdfs` user, under whom HDFS services run by default and who belongs to `dfs.cluster.administrators`.

To obtain a Kerberos ticket for the `hdfs` user:

1. Run the following on the namenode or on the node where you will be running the `haadmin` command:

   ```
   sudo -u hdfs kinit -kt /path/to/keytab/<file>.keytab
    hdfs/<FQDN_OF_THE_NODE>
   ```

2. After obtaining the ticket, verify it using `klist`:

   ```
   sudo -u hdfs klist
   ```

For example, a sample run on a host with FQDN of `node1.phddev.local` using the `hdfs.service.keytab` file will return the following results:

```
[root@node1 ~]# sudo -u hdfs kinit -kt /etc/security/phd/keytab/
hdfs.service.keytab hdfs/node1.phddev.local
[root@node1 ~]# sudo -u hdfs klist
Ticket cache: FILE:/tmp/krb5cc_501
Default principal: hdfs/node1.phddev.local@PHDDEV.LOCAL
Valid starting     Expires            Service principal
09/10/14 22:09:59  09/11/14 22:09:59  krbtgt/PHDDEV.LOCAL@PHDDEV.LOCAL
 renew until 09/17/14 22:09:59
[root@node1 ~]# sudo -u hdfs hdfs haadmin -failover nn1 nn2
```

Uninstalling a Cluster

Use `icm_client uninstall` to uninstall a cluster.

You must run the `icm_client stop` command to stop running clusters before running the `icm_client uninstall` command. You must also ensure that HAWQ has been stopped before uninstalling a cluster.

You will be prompted as to whether you want to preserve the history metrics of the cluster; the default behavior is to preserve the history.

Note: Running `icm_client uninstall` does not delete `dfs.data.dir`, `dfs.name.dir`, `dfs.mapred.dir` and `dfs.checkpoint.dir` directories. This is intentional behavior and preserves user data.

Syntax

```
[gpadmin]# icm_client uninstall -h
Usage: icm_client uninstall [options]

Options:
  -h, --help            Show this help message and exit
  -v, --verbose         Increase output verbosity
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        The name of the cluster to be uninstalled
```

Examples

```
icm_client uninstall -l <CLUSTERNAME>
```

Managing HAWQ

This section describes HAWQ administrative tasks you can perform via the CLI.

Initializing HAWQ

Initializing HAWQ performs the following tasks:

- Initializes the HAWQ master and the segment hosts.
- Starts the HAWQ master, segments, and the underlying postgres database.

You need to initialize HAWQ only once, after the cluster has started and after HDFS is up and running.

Note: Verify that the `postgres` user exists. If it does not, you may have to create it and add it into the `hadoop` group.

To initialize HAWQ:

1. **Security:** If you have deployed a secure cluster with Kerberos authentication, you must create a Kerberos principal for `gpadmin` and run `kinit` before running the next command.

   Note: If you have not deployed a secure cluster, skip this task.

   To add a principal for `gpadmin`:

   a. On the PCC Admin node, run:

   ```
   $ sudo kadmin.local
   $ add princ gpadmin
   $ exit
   ```

   Note: Provide a password for the `gpadmin` principal when prompted.

   b. Run:

   ```
   $ kinit
   ```

2. Verify HDFS is running.

   To verify HDFS is running, log in to the client node, NameNode or DataNode as `gpadmin` and run:

   ```
   $ hdfs dfs -ls /
   ```

   Sample Output:

   ```
      Found 4 items
      drwxr-xr-x   - mapred hadoop          0 2013-06-15 15:49 /mapred
      drwxrwxrwx   - hdfs   hadoop          0 2013-06-15 15:49 /tmp
      drwxrwxrwx   - hdfs   hadoop          0 2013-06-15 15:50 /user
      drwxr-xr-x   - hdfs   hadoop          0 2013-06-15 15:50 /yarn
   ```

3. **Security:** If you have deployed a secure cluster with Kerberos authentication:

Note: If you have not deployed a secure cluster, skip this task.

a. Locate the HAWQ data directory:

   i. On the HAWQ master, open `/etc/gphd/hawq/conf/gpinitsystem_config`.

   ii. Locate `DFS_URL` and obtain the directory after `nameservice` or `namenode`. By default the value of this is `hawq_data`. We will refer to it as <*HAWQ_DATA_DIR*> for the purpose of this document.

b. Create <*HAWQ_DATA_DIR*> on HDFS:

   i. Start the cluster using `icm_client`.

   ii. Make sure HDFS service is up and running.

   iii. As `gpadmin`, on the namenode or client machine, run:

```
kinit
hadoop fs -mkdir /<HAWQ_DATA_DIR>
hadoop fs -chown -R postgres:gpadmin /<HAWQ_DATA_DIR>
hadoop fs -mkdir /user/gpadmin
hadoop fs -chown gpadmin:gpadmin /user/gpadmin
hadoop fs -chmod 777 /user/gpadmin
kdestroy
```

4. As `gpadmin`, exchange keys, then initialize HAWQ from the HAWQ master.

   Note: `ssh` to the HAWQ Master before you initialize HAWQ.

   For example:

```
$ su - gpadmin
$ source /usr/local/hawq/greenplum_path.sh
$ gpssh-exkeys -f HAWQ_HOSTS.txt # where HAWQ_HOSTS.txt has a set of hawq
 nodes
$ /etc/init.d/hawq init
```

   Note: You do not need to start HAWQ. It is implicitly started as part of the initialization.

5. If you have a HAWQ Standby master in your cluster configuration, initialize it using `gpinitstandby`:

   a. `gpinitstandby` reads the master data directory location from the `$MASTER_DATA_DIRECTORY` environment variable, so before running `gpinitstandby`, run the following:

```
$ export MASTER_DATA_DIRECTORY=<MASTER_DIRECTORY>/gpseg-1
```

   For example:

```
$ export MASTER_DATA_DIRECTORY=/data0/master/gpseg-1/gpseg-1
```

   b. Then, still as `gpadmin`, initialize the Standby master:

```
$ gpinitstandby -s <HAWQ_STANDBY_MASTER_FQDN>
```

Note: Hive with HAWQ/PXF

If you are planning to configure Hive with HAWQ/PXF, check that the Hive Metastore service is available and running (anywhere on the cluster) and that you have set the property `hive.metastore.uri` in the `hive-site.xml` file on the NameNode to point to that location.

Starting HAWQ

Note that starting and stopping HAWQ can only be initiated directly on the HAWQ Master. More information about HAWQ can be found in the *HAWQ Installation Guide* and the *HAWQ Administrator Guide*.

Run the `start` command to start up the HAWQ master and all the segments hosts, including the postgres database.

```
[gpadmin]# /etc/init.d/hawq start
```

Note: If you are initializing HAWQ, you do not need to perform this task. It is implicitly done during HAWQ Initialization.

Stopping HAWQ

Note that starting and stopping HAWQ can only be initiated directly on the HAWQ Master. More information about HAWQ can be found in the *HAWQ Installation Guide* and the *HAWQ Administrator Guide*.

Run the `stop` command to stop the HAWQ master, segments hosts, and the postgres database on the HAWQ master.

```
[gpadmin]# /etc/init.d/hawq stop
```

Modifying HAWQ User Configuration

If you are using Pivotal Command Center, you must modify your HAWQ user configuration file.

This is because the Admin host is not part of the HAWQ cluster. Modifying the `pg_hba.conf` file on the HAWQ Master host gives the Admin host the ability to remote query HAWQ .

1. Log in to the HAWQ Master as user `gpadmin`.
2. In the `$MASTER_DATA_DIRECTORY/pg_hba.conf` file (the location of the HAWQ Master Directory is defined in the `<hawq.master.directory>` section of the `clusterConfig.xml` file used for deployment of the Cluster):
   a. Find the entry:

      ```
      host all gpadmin <master_host_ip>/32 trust
      ```

   b. Change the subnet entry, depending on your network configuration:

      ```
      host all gpadmin <master_host_ip>/24 trust
      ```

3. Restart HAWQ:

   ```
   /etc/init.d/hawq restart
   ```

Run the following command to test HAWQ from the Admin host:

```
$ sudo -u gpadmin psql -h <HAWQ_MASTER_NODE> -p <HAWQ_PORT> -U gpadmin
 postgres -c "select * from pg_stat_activity;"
```

Expanding HAWQ

HAWQ Segments can be expanded.

Before you expand a HAWQ segment, you need to add slaves to the cluster by either:

- Running the `add-slaves` command (see *Expanding a Cluster* on page 118).
- Manually editing the `hawq-segments` section of the `clusterConfig.xml` file, then running the `reconfigure` command (see *Reconfiguring a Cluster* on page 115).

Once you have added the slaves, you can then expand HAWQ using the `gpexpand` command. See the *HAWQ Administration Guide: Expanding the HAWQ System* for details.

Managing PHD Roles and Hosts

Pivotal HD supports starting or stopping entire clusters or individual roles on a selected hosts. If you want to start and stop the roles manually, follow these steps:

You have two options when managing cluster and individual roles.

- *Managing Locally* on page 146
- *Managing Remotely* on page 146

Managing Locally

You can manage the service role on the target host locally. For example, to restart the DataNode:

```
node100:gpadmin# ssh gpadmin@node100
gpadmin# sudo service hadoop-hdfs-namenode restart
```

Managing Remotely

You can manage the service role remotely across one of the target hosts. For example, to restart the DataNode:

```
node100.gpadmin# massh node100 verbose 'sudo service hadoop-hdfs-datanode
 restart'
```

To restart all the DataNodes remotely, create a newline-separated file named `hostfile` that contains all the DataNodes to **start**, **stop**, **restart**, or **check** status.

```
gpadmin# massh hostfile verbose 'sudo service hadoop-hdfs-datanode restart'
```

Pivotal HD Services Scripts

The following table shows the service commands to **start**, **stop**, **restart**, or **check** status for each service role,.

Role Name	Service Command
NameNode	`sudo service hadoop-hdfs-namenode {starts|stop|status|restart}`
Secondary NameNode	`sudo service hadoop-hdfs-secondarynamenode {starts|stop|status|restart}`
DataNode	`sudo service hadoop-hdfs-datanode {starts|stop|status|restart}`
Resource Manager	`sudo service hadoop-yarn-resourcemanager {starts|stop|status|restart}`
Node Manager	`sudo service hadoop-yarn-nodemanager {starts|stop|status|restart}`
History Server	`sudo service hadoop-mapreduce-historyserver {starts|stop|status|restart}`
Zookeeper Server	`sudo service zookeeper-server {starts|stop|status|restart}`
HBase Master	`sudo service hbase-master {starts|stop|status|restart}`
HBase Region Server	`sudo service hbase-regionserver {starts|stop|status|restart}`
HAWQ Master	`sudo /etc/init.d/hawq {starts|stop|status|restart}`
Quorum Journal node	`sudo /etc/init.d/hadoop-hdfs-journalnode {start|stop|status|restart}`

PHD Services Reference

Overriding Directory Permissions

The following table shows the list of directories that Pivotal HD overrides with specific ownership and permissions.

Directories not mentioned in the below list follow standard Apache ownership and permission convention.

On the Local Filesystem

Service	Directory	Location	Owner	Permissions
HDFS	**hadoop.tmp.dir**	All Hadoop nodes	**hdfs:hadoop**	777
	dfs.namenode.name.dir	NameNode	**hdfs:hadoop**	700
	dfs.datanode.data.dir	DataNodes	**hdfs:hadoop**	770
	dfs.namenode.checkpoint.dir	Secondary NameNode	**hdfs:hadoop**	700
	dfs.journalnode.edits.dir	Journal Node	**hdfs:hadoop**	755
YARN	**mapreduce.cluster.local.dir**	All yarn nodes	**mapred:hadoop**	755
	mapreduce.cluster.temp.dir	All yarn nodes	**mapred:hadoop**	755
	yarn.nodemanager.local-dirs	Node Managers	**yarn:yarn**	755
	yarn.nodemanager.log-dirs	Node Managers	**yarn:yarn**	755
ZooKeeper	**dataDir (/var/lib/zookeeper)**	Zookeeper Servers	**zookeeper:zookeeper**	775
	dataDir/myid	Zookeeper Servers	**gpadmin**	644
HAWQ	**MASTER_DIRECTORY**	HAWQ Master & Standby	**gpadmin:hadoop**	755
	DATA_DIRECTORY	HAWQ Segments	**gpadmin:hadoop**	755

On HDFS

Service	Directory	Owner	Permissions
HDFS	**hadoop.tmp.dir**	**hdfs:hadoop**	777
	/tmp	**hdfs:hadoop**	777
	mapreduce.jobtracker.system.dir	**mapred:hadoop**	700

Service	Directory	Owner	Permissions
	yarn.app.mapreduce.am.staging-dir (/user)	**mapred:hadoop**	777
	mapreduce.jobhistory.intermediate-done-dir (/user/history/done)	**mapred:hadoop**	777
	mapreduce.jobhistory.done-dir (/user/history/done)	**mapred:hadoop**	777
	yarn.nodemanager.remote-app-log-dir	**mapred:hadoop**	755
HBase	**hbase directory (/apps/hbase/data)**	**hdfs:hadoop**	775
HAWQ	**hawq directory (/hawq_data)**	**hdfs:hadoop**	755

Pivotal HD Users and Groups

Service	Users	Group	Login
PHD	gpadmin	gpadmin	Yes
HDFS	hdfs	hadoop	Yes
MapReduce	mapred	hadoop	Yes
Hbase	hbase	hadoop	No
Hive	hive	hadoop	No
Zookeeper	zookeeper	zookeeper	No
Yarn	yarn	yarn	No
PHD, HAWQ	postgres	postgres	Yes
Puppet	puppet	puppet	No

Pivotal HD Ports

Note: If you are running a firewall, ensure all ports are open.

Component	Service	Port	Protocol	Access	Configuration Parameters
HDFS	NameNode Metadata Service	8020	IPC	External	fs.defaultFS
	NameNode Web UI	50070	HTTP	External	dfs.namenode.http-address
	Secondary NameNode Web UI	50090	HTTP	Internal	dfs.namenode.secondary.http address
		50495	HTTPS	Internal	dfs.secondary.https.address

Component	**Service**	**Port**	**Protocol**	**Access**	**Configuration Parameters**
	DataNode Data Transfer	50010 (non-secure mode)		External	dfs.datanode.address
		1004 (secure mode)			
	DataNode Metadata Operations	50020	IPC	External	dfs.datanode.ipc.address
	DataNode HTTP/HTTPS Address	50075 (non-secure mode)	HTTP	External	dfs.datanode.http.address
		1006 (secure mode)	HTTP		
		50475	HTTPS	External	dfs.datanode.https.address
	HDFS NFS server	2049			nfs3.server.port
	HDFS NFS mount daemon	4242			nfs3.mountd.port
	HDFS Backup Node Server	50100			dfs.namenode.backup.addres
	HDFS Backup Node Server HTTP	50105	HTTP		dfs.namenode.backup.http-address
	Quorum Journal node port	8485		Internal	dfs.journalnode.rpc-address
	Quorum Journal Node Web UI	8480	HTTP	Internal	dfs.journalnode.http-address
YARN	ResourceManager Web UI	8088	HTTP		yarn.resourcemanager.webap
		8090	HTTPS		yarn.resourcemanager.webap
	NodeManager Web UI	8042	HTTP		yarn.nodemanager.webapp.ad
		8044	HTTPS		yarn.nodemanager.webapp.ht
	ResourceManager	8030			yarn.resourcemanager.schedu
		8031			yarn.resourcemanager.resour tracker.address
		8032	IPC		yarn.resourcemanager.addres
		8033			yarn.resourcemanager.admin.
	NodeManager Localizer	8040	IPC		yarn.nodemanager.localizer.a
HBASE	HBase Master	60000		External	hbase.master.port
	HBase Master Web UI	60010	HTTP	External	hbase.master.info.port

Component	Service	Port	Protocol	Access	Configuration Parameters
	HBase RegionServer	60020		External	hbase.regionserver.port
	HBase RegionServer Web UI	60030	HTTP	External	hbase.regionserver.info.port
	HBase REST Server	8080	HTTP	External	hbase.rest.port
	HBase REST Server Web UI	8085	HTTP	External	hbase.rest.info.port
	HBase ThriftServer	9090		External	Pass -p <port> on CLI
	HBase ThriftServer Web UI	9095	HTTP	External	hbase.thrift.info.port
	HQuorumPeer	2181			hbase.zookeeper.property.clie
	HQuorumPeer	2888			hbase.zookeeper.peerport
	HQuorumPeer	3888			hbase.zookeeper.leaderport
ZOOKEEPER	ZooKeeper Server	2181		External	zoo.cfg - clientPort
	ZooKeeper Peers	2888		Internal	zoo.cfg - X in server.N=hostN:X:Y
	ZooKeeper Leader	3888		Internal	zoo.cfg - Y in server.N=hostN:X:Y
HIVE	Hive Server	10000		External	hive-env.sh - HIVE_PORT
	Hive Metastore	9083		External	hive.metastore.uris
	Hive Web Interface	9999	HTTP		hive.hwi.listen.port
	Hive Server2 Thrift	10000		External	hive.server2.thrift.port
	Hive Server2 Thrift HTTP	10001	HTTP	External	hive.server2.thrift.http.port
HCatalog	HCatalog	9083			
	Web HCatalog	50111			
OOZIE	Oozie Server	11000	HTTP	External	oozie-env.sh - OOZIE_HTTP_PORT
		11443	HTTPS	External	oozie-env.sh - OOZIE_HTTPS_PORT
	Oozie Server Admin	11001			oozie-env.sh - OOZIE_ADMIN_PORT

Component	Service	Port	Protocol	Access	Configuration Parameters
SQOOP	Sqoop Metaserver	16000		External	sqoop.metastore.server.port
HAWQ	HAWQ Master	5432			
	HAWQ Port Base	40000 This port number increases by 1 for every segment on each host. If you have three segments per host, it would be 40000, 40001, and 40002 across all HAWQ segment servers.			
KDC	Kerberos KDC Server	88			
	ssh	22			

Chapter 10

PHD Frequently Asked Questions (FAQ)

Can I deploy multiple clusters from the same admin?

Yes, you can deploy any number of Pivotal HD clusters from the same admin. You must deploy them in succession, not simultaneously.

Can I modify the topology (host to role mapping) of the cluster after the initial install?

Yes, you can change slaves' roles using the CLI, but the master role must be changed manually. If you want to change the master role, contact Support.

How do I reformat the namenode?

Important: These steps will erase all data on HDFS.

As user `hdfs`:

1. On the namenode, clean up the data in the directories specified for `dfs.datanode.name.dir`.
2. On all the datanodes, clean up the data in the directories specified for `dfs.datanode.data.dir`.
3. On the namenode, run:

```
hadoop namenode format -force
```

Certain services such as hadoop-hdfs-namenode or hadoop-hdfs-datanode do not come up when I run "start cluster"?

Refer to Debugging tips in the Troubleshooting section. It may be that the ports being used by the specific service are already in use. Verify whether the port is already being used using `-netstat -na`. Kill the existing process if necessary

What group and users are created by Pivotal HD?

Please refer to the Troubleshooting section for details about the users and directories created by PCC.

What is the allowed time difference amongst the cluster nodes versus the admin node?

The allowed time difference between the cluster nodes is +/-60 secs of admin node time. If the time difference is more, the SSL authentication might fail, leading to cluster deployment failures.

Does PCC support simultaneous deployment of multiple clusters?

No. Concurrent deployment is not allowed. Please wait till the first deployment is complete before starting another.

Does PCC support hostname both in IP address and FQDN format?

No, only FQDN format is currently supported.

Can a node be shared between different clusters?

No, nodes cannot be shared between clusters.

I installed puppet-2.7.20 from the Puppet Labs repository, but Pivotal HD does not work?

Pivotal HD requires the version of puppet shipped with the product and not the downloadable version from the Puppet Labs repository. Uninstall Puppet and install the one shipped with the product using the `icm_client preparehosts` command.

How do I clean up the nodes if a cluster deployment fails?

Uninstall the cluster using the `icm_client uninstall` command, then follow the instructions for deploying the cluster again.

Will I lose my data if I uninstall the cluster?

Uninstalling the cluster will not wipe out any data. But a subsequent installation would wipe out the configured mount points upon confirmation. Make sure you back out the data.

Will I lose my data if I upgrade the PHD/ADS stack through the stack import utility?

Upgrading any stack using the import utility will not affect your cluster/data as long as the upgrade is compatible with the existing data layout.

Can I upgrade Pivotal Command Center while the clusters are functioning?

Yes you can. Upgrading the Admin node will not interfere with any of the clusters.

How do I change the port used by Pivotal HD?

1. Log onto the machine as `root`. 2. Stop Pivotal Command Center:

   ```
   service commander stop
   ```

2. Change the port in the jetty file, say from 8080 to 8085:

   ```
   Update the JETTY_PORT property to 8085 in: /usr/lib/gphd/gphdmgr/bin/setenv.sh
   Update ICM_URL property to 8085 in /etc/gphd/gphdmgr/conf/gphdmgr.properties
   Update the gphdmgr_port to 8085 in /usr/local/greenplum-cc/config/app.yml

   \#Replace 8080 with 8085 in the following files
   sed \-i 's/8080/8085/g'  /usr/lib/gphd/gphdmgr/lib/client/InputReaders.py
   sed \-i 's/8080/8085/g' /usr/lib/gphd/gphdmgr/lib/client/GPHDSync.py
   sed \-i 's/8080/8085/g' /usr/lib/gphd/gphdmgr/lib/client/WSHelper.py
   ```

3. Start Pivotal Command Center again:

   ```
   service commander start
   ```

Chapter

11

PHD Troubleshooting

This section provides common errors you may receive and how to troubleshoot or workaround those errors.

Debugging Errors

Pivotal Command Center has many different log files. Finding the exact log may initially be challenging at the beginning.

Here is a quick guide on how to identify the issues:

Pivotal HD Installation

All installation errors will be logged under: `/var/log/gphd/gphdmgr/installer.log`

Cluster Deployment

If you see a 500 Internal Server Error, check the following logs for details: `/var/log/gphd/gphdmgr/gphdmgr-webservices.log`

If you see Puppet cert generation errors, check **`/var/log/gphd/gphdmgr/gphdmgr-webservices.log`**

If config properties are not making it into the cluster nodes, check `/var/log/gphd/gphdmgr/gphdmgr-webservices.log`

If you see `GPHDClusterInstaller.py` script execution error, check `/var/log/gphd/gphdmgr/GPHDClusterInstaller_XXX.log`

Sometimes `/var/log/messages` can also have good information, especially if the deployment fails during the puppet deploy stage.

In general if something fails on the server side, look at the logs in this order:

- /var/log/gphd/gphdmgr/gphdmgr-webservices.log
- /var/log/gphd/gphdmgr/GPHDClusterInstaller_XXX.log
- /var/log/messages

Cluster Nodes Installation

If there are no errors on the admin side, but the installation failed on the cluster nodes, check the latest log file: `/tmp/GPHDNodeInstaller_XXX.log`

Search for the first occurrence of the word `merr;` that will point to the most probable issue.

Services Start

Check for the corresponding log file under `/var/log/gphd/` directory.

For example, if the namenode doesn't start, look at the `/var/log/gphd/hadoop-hdfs/hadoop-hdfs-namenode-hostname.log` file for details.

Puppet SSL Errors

For errors like: `"Unable to generate certificates"` `"SSLv3 authentication issues on the client"`

As `root`, do the following:

Ensure the hostname on all machines is a fully qualified domain name. (see the `HOSTNAME` field in `/etc/sysconfig/network`.)

Run:

```
service commander stop
```

On **all machines including cluster nodes**, run:

```
rm -rf /var/lib/puppet/ssl-icm/*
```

On the admin node, ensure there is no puppet master process running by running:

```
ps ef | grep puppet
```

If there is, `kill -9` any running puppet process:

```
ps -ef|grep puppet|awk '{print $2}'|xargs kill -9
```

Make sure there are no certificates listed by running:

```
puppetca list --all
```

You can run `puppetca clean --all` to clean any certificates

Restart the puppet master:

```
service puppetmaster start
```

Verify there is just one certificate:

```
puppetca list --all
```

Stop the puppet master and start nmon:

```
service puppetmaster stop
service commander start
```

Now retry your deployment.

Upgrade/Reconfigure Errors

Following an upgrade of Command Center, unable to Start/Stop cluster with invalid hostnames

This is because there is now a check for invalid characters in cluster names.

Workaround: First reconfigure the cluster to a different name:

```
icm_client reconfigure -l <old_cluster_name> -c <config directory with new
 clustername>
```

Then try starting/stopping the cluster:

```
icm_client start -l <cluster_name>
icm_client stop -l <cluster_name>
```

Other Upgrade/Reconfigure Errors

After upgrading PHD stack from 1.0.2 to 1.0.3 release, hbase master fails to start if hbase-master is not co-located with either namenode or datanode.

Workaround: On hbase-master node, run: `yum upgrade hadoop-hdfs`. Go to the `/usr/lib/gphd` directory. Point the `hadoop-hdfs` symlink to the newer `hadoop-hdfs` version.

If you see a `hostRoleMapping should not be changed for other services` error, make sure the `clusterConfig.xml` file has not been changed for any of the already existing services. Even if it is the same set of hosts, but in a different order, make sure you maintain the order in the comma separated list.

If you see `ERROR:Fetching hadoop rpm name on namenode: <`**`host`**`> failed` error, it is most likely a case where the cluster was being upgraded from 1.0.0 to 1.0.2 and there was an error during upgrade.

Workaround: Run `yum install hadoop-2.0.2_alpha_gphd_2_0_1_0-14.x86_64` on the namenode and retry upgrade.

If you are upgrading a cluster with HBase, Hive, or PXF configured as a service, you must manually reinstall those services. See *Upgrading PHD 2.0.x to 2.1.0* on page 78 for details.

HA-related Errors

If the cluster fails to start with HA enabled:

- Check the status of the journal node (`/etc/init.d/hadoop-hdfs-journalnode` status) on all hosts and ensure they are running.
- Check if the "namenode" (configured as `namenodeid1` in `clusterconfig.xml`) is formatted and successfully started. Be sure to check `/var/log/gphd/gphdmgr/gphdmgr-webservices.log` and, if needed, the namenode logs on the namenode host: `/usr/lib/gphd/hadoop/logs/hadoop-hdfs-namenode*log`
- Check if the "standbynamenode" (configured as `namenodeid2` in `clusterconfig.xml`) is formatted and successfully started. The namenode logs should have details on any errors, if the standbynamenode failed to format or start.
- If standbynamenode fails to start because it is not formatted and restarting the cluster does not format the name node, please contact support team for help.
- If you are converting a non-HA cluster to HA, please follow the documented steps. It is important to start the journal nodes and initialize the edit logs from the namenode of the existing cluster before starting the cluster.

Other Errors

Command Center Installation fails due to failed dependencies

If, during the installation of PCC, you receive a facter mismatch error like the following:

```
PCC-2.2.0-175]# rpm -ev facter
error: Failed dependencies:
facter >= 1.5 is needed by (installed) puppet-2.7.9-1.el6.noarch
```

Remove facter using the command:

```
yum erase facter
```

Then run the PCC installation again.

Cluster Deployment fails due toRPM Dependencies

Ensure that the base OS repo is available. You might have to mount the CD that comes with the OS installation or point yum to the correct location, such as the NFS mount point on all the cluster nodes.

Unable to access the Namenode Status Web page

If the host returns a short hostname instead of FQDN for `hostname()`, it is possible that the namenode status link cannot be accessed from external networks.

The solution is to either ensure that the `hostname()` returns FQDN on the namenode host, or change the `dfs.http.address` value to `0.0.0.0` in the `hdfs-site.xml` and restart namenode.

```
<property>
<name>dfs.http.address</name>
<value>0.0.0.0:50070</value>
</property>
```

Installation Fails due to Directory Permissions

Check if the umask is set to 0022. If not, set the umask in the `.bashrc` as "umask 0022", then retry the PCC installation.

Deployment Fails due to Problems with YUM Repository

Verify that the admin node is reachable from the agent node.

If you have configured proxy servers, refer to the section titled *Working with Proxy Servers*.

Installation Fails due to Problems with the SSL certificate

Check if `dnsdomainname` returns an empty value. If yes, you need to ensure that the `dnsdomainname` returns the correct domain.

Cluster Node Installation Failurewithout Generating aLog File

Ensure that passwordless ssh is setup between the admin node and the cluster nodes.

Ensure that the puppet, facter and ruby rpms are the same as that on the admin node

Ensure that the user `gpadmin` has sudo and no requiretty access on the cluster node (check for the existence of file: `/etc/sudoers.d/gpadmin`)

Then, retry the deployment.

Puppet certificate failure

Follow the instructions in the *Puppet SSL Errors* section.

Package Bundle Not Found

If you sudo into the system as root, ensure that you sudo with the environment. That is: `sudo su -` Do not forget the hyphen at the end.

If you directly login as root with the password and you still see the above issue, check if the /usr/local/bin/bundle exists. If not, build it:

```
gem install bundler
```

Add /usr/local/bin to PATH, regardless:

```
[]# vi ~/.bashrc
```

Append export PATH=$PATH:/usr/local/bin, then save

```
[]# source ~/.bashrc
```

Cluster Deployment Fails due to Missing Packages

The above error can be identified by following the instructions on *Cluster Nodes Installation* errors section above.

Install **nc** and **postgres-devel** packages on all the cluster nodes or point them to a repo that contains the rpms.

Working with Proxy Servers

It is sometimes required that all outgoing http traffic use a HTTP proxy. PCC installer sometimes pulls rpms from an external repos such as an EPEL6 repo if the external repos are configured and if any packages are missing on the host.

If you configure the proxy settings in /etc/yum.conf for the cluster node, cluster deployments might fail because yum will send all gphd.repo requests to the proxy, which in turn will fail to connect to the admin node's local repo.

Here are a few workarounds:

Workaround 1:

- Remove the proxy settings from yum.conf and
- Make sure following params are set in ~root/.bashrc

  For example: export http_proxy=http://proxy:3333 export no_proxy=local.domain ## this is the local domain for hadoop cluster
- Modify these files so gphd.repo gets pushed out with a FQDN name instead of shortname: /etc/puppet/modules/yumrepo/templates/yumrepo.erb

  Change from:

  ```
  baseurl=http://<%= scope.lookupvar("params::config::admin_host")  %>/<%=
   scope.lookupvar("params::config::repopath") %>
  ```

  Change to:

  ```
  <replace node.full.domain.com> with the FQDN of the admin node
  baseurl=http://node.full.domain.com/<%=
   scope.lookupvar("params::config::repopath") %>
  ```

Workaround 2:

- Enable NFS and export /usr/lib/gphd/rpms to all cluster nodes.
- Mount the nfs repo on all cluster nodes:

  ```
  mount gpcc:/usr/lib/gphd/rpms /local_repo
  ```

- Modify these files: `/etc/puppet/modules/yumrepo/templates/yumrepo.erb`

  Change from:

  ```
  baseurl=http://<%= scope.lookupvar("params::config::admin_host")  %>/<%=
   scope.lookupvar("params::config::repopath") %>
  ```

  Change to:

  ```
  baseurl={nolink:file:///local_repo/}
  ```

Capital Letters in Hostname

PCC fails to deploy if the hostnames contain uppercase letters. For example: `Node0781.domain.com`.

Rename the hostname with only lowercase letters before proceeding with the deployment.

Resolving postgres port Conflict Issue

If you face a postgres port conflict or wish to change the default postgres port, follow the steps below:

1. Stop PCC service:

   ```
   root# service commander stop
   ```

2. Add the new port `<hostname>:5435` in the Pivotal HD properties file:

   ```
   vim /etc/gphd/gphdmgr/conf/gphdmgr.properties
   gphdmgr.db.url=jdbc:postgresql://localhost:5435/gp
   ```

3. Change the port number in `postgresql.conf`:

   ```
   vim /var/lib/pgsql/data/postgresql.conf "port = 5435"
   ```

4. Edit the `init.d/postgresql` file:

   ```
   vim /etc/init.d/postgresql
   #Change the PGPORT to 5435 "PGPORT=5435"
   root# service commander start
   ```

Resolving HTTP Port Conflict

Check the FAQ section: How do I change the port used by Pivotal HD?

Errors like Ambit: Push Failed

If you see errors like the following:

```
root# icm_client add-user-gpadmin -f hosts
Ambit : Push Failed
Had : Push Failed
Issues : Push Failed
Generating : Push Failed
A : Push Failed
List : Push Failed
```

This is an ambit bug. If there are hostnames (only the name part, not the domain) that are substrings of other hostnames, then this issue can occur.

For example: `host1.emc.com`, `host11.emc.com`

This error can be ignored for now as the actual deployment still goes through.

Preparehosts Errors Out While Creating gpadmin User

Make sure SELinux needs to be either disabled or in permissive mode for the hosts.

(See the *PCC User Guide* for instructions to disable SELinux.)

HAWQ Initialization Failing

Make sure your cluster is up and running with the Hadoop services, prior to initializing HAWQ (`hawq init`). If the failure still persists, make sure the HAWQ nodes have been prepared (*PHD Install 7 - Deploy the Cluster* on page 39, `icm_client deploy` syntax) to reflect the kernel configurations required for HAWQ. If you still have a problem, you might be running short of the memory required to run HAWQ at scale. Refer to *HAWQ Administration* to configure/modify the system memory requirements.

Installing HAWQ on Dirty Cluster Nodes Previously Configured with HAWQ

If you wish to deploy or initialize HAWQ on:

a) A cluster that had an older uninstalled HAWQ cluster, or

b) A cluster that failed in its attempts to initialize HAWQ

You will need to perform the following steps before initializing HAWQ with the new cluster nodes:

1. Ensure that `HAWQ_Hosts.txt` contains all the HAWQ hosts that you want to clean up.
2. Run the following command against each DIRECTORY configured in `<hawq.segment.directory>` and in `<hawq.master.directory>` in the cluster configuration (`clusterConfig.xml`)

```
gpadmin# massh HAWQ_Hosts.txt verbose 'sudo rm -rf DIRECTORY/*'
```

The above command cleans up the stale HAWQ master and segment data directory contents.

Errors Related to VM Memory

If you are planning to deploy a HAWQ cluster on VMs that have memory limits lower than the optimized/recommended requirements, you might encounter `Could not create the Java virtual machine` type errors. In these cases, you can reconfigure memory usage, as follows:

- Prior to running the prepare HAWQ utility, open the `/usr/lib/gphd/gphdmgr/hawq_sys_config/sysctl.conf` file and change the value of the following parameter from 2 to 0:

  ```
  vm.overcommit_memory =2
  ```

- In the `clusterConfig.xml`, update `<hawq.segment.directory>` to include only one segment directory entry (instead of the default 2 segments).

Chapter 12

PHD REST API

This section describes how to access PHD's REST API via Swagger.

Note: The HAWQ API is currently being re-developed. As of now this API is not functional.

Swagger with OAuth

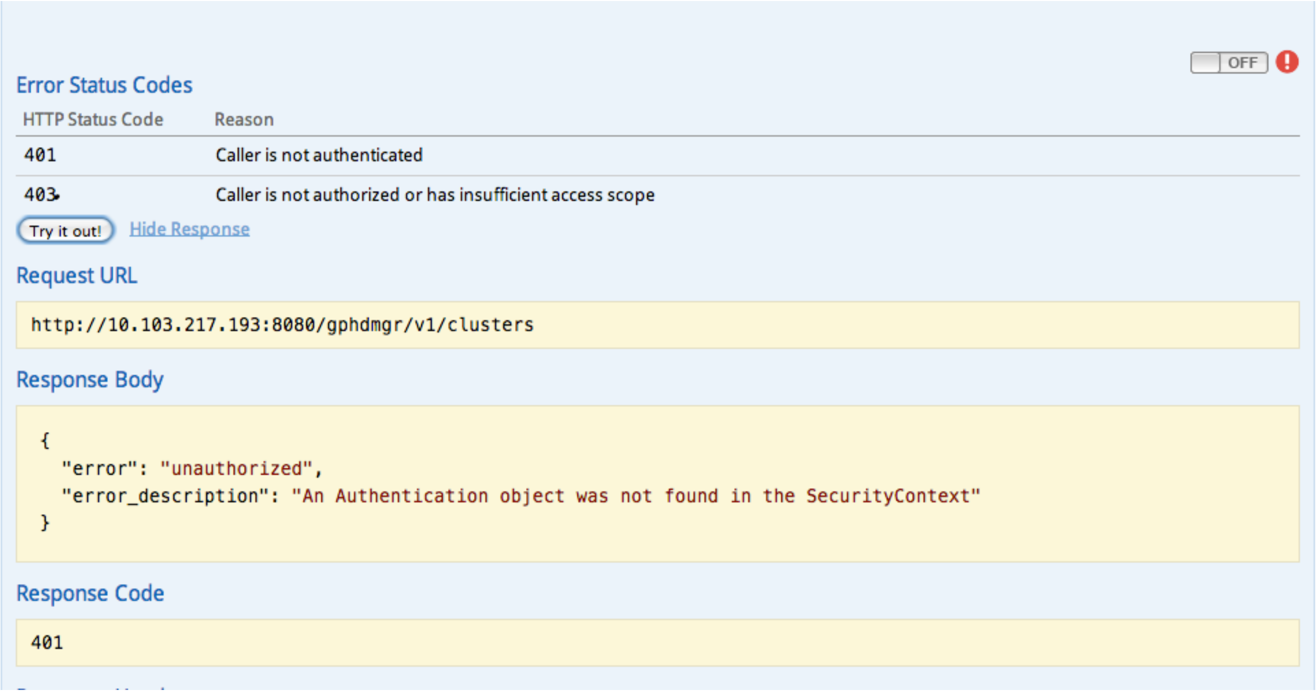
This topic contains instructions for using Swagger for PHD APIs using OAuth authentication.

- Swagger is a specification and complete framework implementation for describing, producing, consuming, and visualizing RESTful web services. The Swagger UI allows you to interact with the API in a sandbox UI.
- OAuth is an open standard for authentication.

To use the Swagger API with OAuth:

1. Go to: `https://<hostname>:8080/gphdmgr/api`

   The following Swagger UI appears:



Error Status Codes

HTTP Status Code	Reason
401	Caller is not authenticated
403	Caller is not authorized or has insufficient access scope

Request URL

```
http://10.103.217.193:8080/gphdmgr/v1/clusters
```

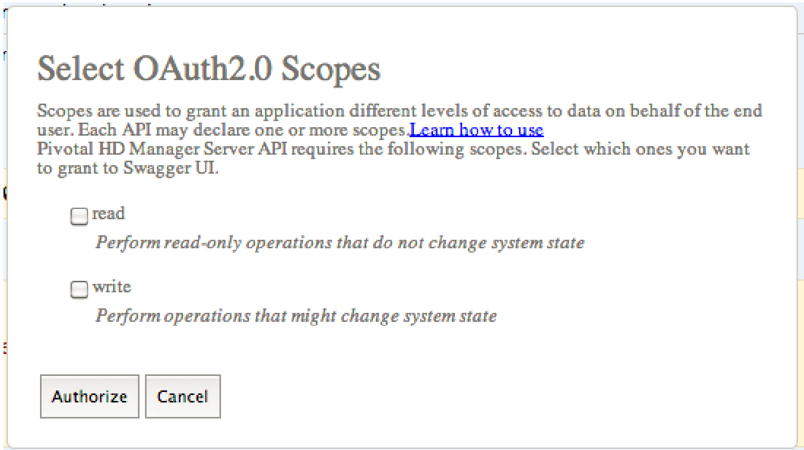
Response Body

```
{
  "error": "unauthorized",
  "error_description": "An Authentication object was not found in the SecurityContext"
}
```

Response Code

```
401
```

2. Click **OFF** (upper right of screen).
3. You are prompted to **Select OAuth2.0 Scopes**.



Select OAuth2.0 Scopes

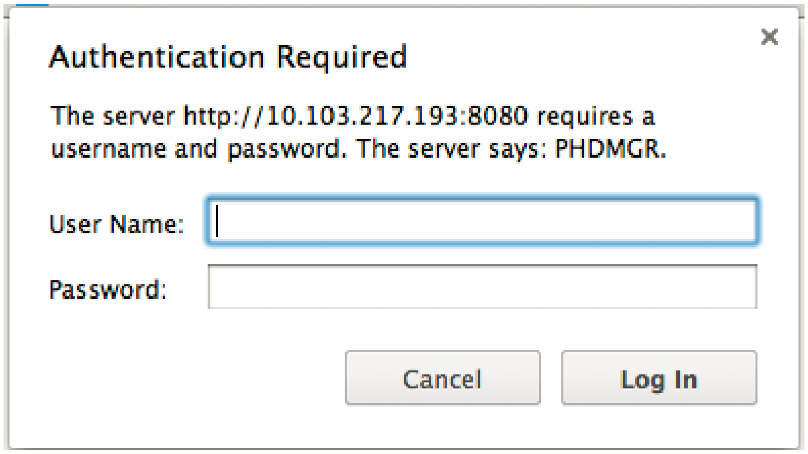
Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes. Learn how to use
Pivotal HD Manager Server API requires the following scopes. Select which ones you want to grant to Swagger UI.

☐ read
Perform read-only operations that do not change system state

☐ write
Perform operations that might change system state

Check both the **read** and **write** boxes, then click **Authorize**.

4. You are prompted "Authentication Required" and you need to log in:



Authentication Required

The server http://10.103.217.193:8080 requires a username and password. The server says: PHDMGR.

User Name:

Password:

5. On the Admin node, locate the password for `gpadmin` from `/usr/local/pivotal-cc/config/oauth2-users.conf`. For example:

```
[root@centos65-1 config]# cat /usr/local/pivotal-cc/config/oauth2-
users.conf
gpadmin=jQk39cbeTx60o3kgeI-7hw,ROLE_USER,enabled
```

6. Enter the username (`gpadmin`) and password you just retrieved into the "Authentication Required" prompt.
7. After login, you can click **Try it Out** in the Swagger UI and you will get a successful response code.

PHD REST API List

Note: The API may have changed since the release. Always refer to the latest list at: `https://<hostname>:8080/gphdmgr/api`

API	Short Description
isi-hdfs : Isilon HDFS API	ISILON status
system : System data API	Get System Data
zookeeper : Zookeeper Service API	Zookeeper status/start/stop

API	Short Description
hive : Hive Service API	Hive status/start/stop
hadoop_command : Hadoop Command API	Set safemode to namenode
v1 : ICM WebServices API	Perform actions on cluster or get information
hawq : HAWQ API	HAWQ status
hbase : HBase API	Hbase status, metrics, and actions (start, stop)
mapreduce : MapReduce API	MapReduce job status and usage
apps : App API	Information about Yarn apps and jobs
queues : Queue API	Job queue status and metrics
jobs : Job API	Jobs-related information
hdfs : HDFS API	HDFS-related actions and information
admin : Admin API	Cluster deployment-related actions
yarn : Yarn service API	Yarn-related information
logs : NODE AGENT API	Logs search/content retrieval