Pivotal™

PRODUCT DOCUMENTATION

# Pivotal HD Enterprise

Version 2.0

# Installation and Administrator Guide

Rev: A03 – May 7, 2014

# Copyright

**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://gopivotal.com/about-pivotal/support.

# Contents

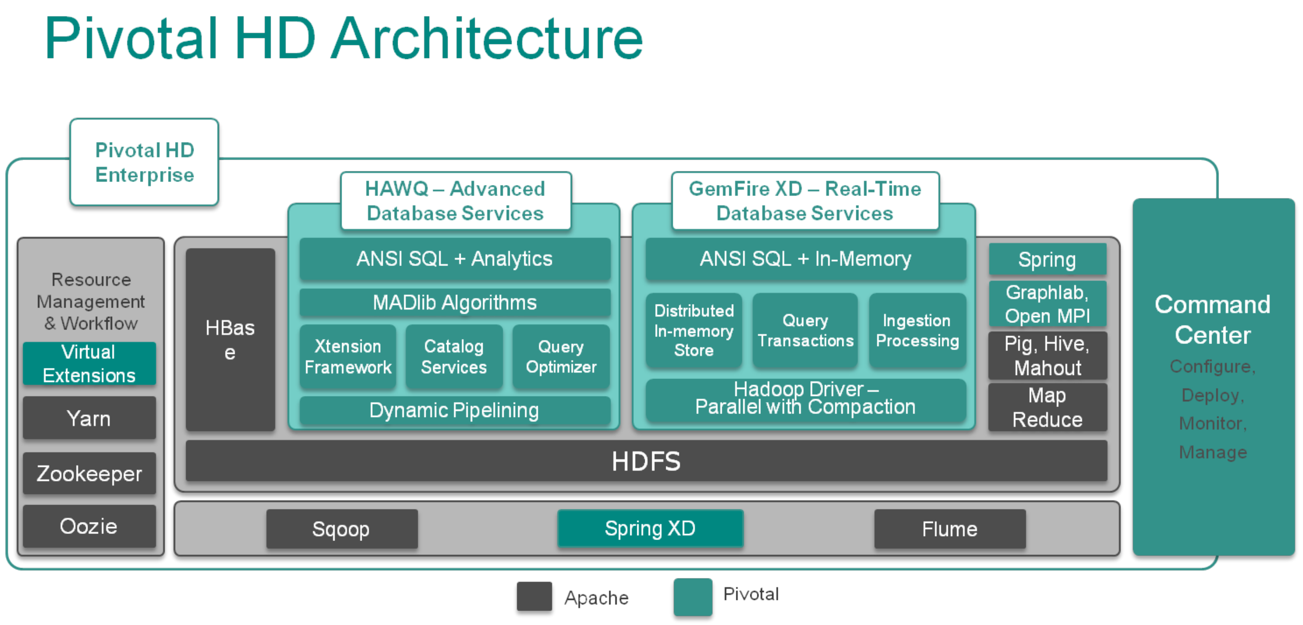
# Chapter 1 Overview of PHD

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

**Topics:**

## 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 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**
    - **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** - Pivotal Command Center (PCC) Is a Web-based interface for configuration and deployment of clusters, and for monitoring & management of a Pivotal HD 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.

  Pivotal Command Center (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). You can start and stop the clusters using either the CLI or the GUI.

  ⚠ 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.
- **PRTS - Pivotal Real Time Services** - Pivotal HD 2.x includes support for GemFire XD (GFXD), an offering of PRTS. For further information about GemFire XD installation and configuration; refer to the section Configuring GemFire XD .
- **Hamster** - Developed by Pivotal, Hamster is a framework which enable users running 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 Pivotal Command Center CLI, or manually.

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

The following services can only be deployed and configured manually (see the *Pivotal HD Enterprise 2.0 Stack and Tool Reference Guide* 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. |

| Role Name | Description |
|---|---|
| DataNodes | A DataNode stores data in the HDFS. A functional filesystem has more than one DataNode, with data replicated across all nodes. |
| 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. |

**Notes**

- 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 Pivotal Analytic Database (Greenplum) 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.

## 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 running 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.

**Topics:**

## 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.<br>• Reconfigures the supported cluster services.<br>• Multi-cluster support.<br>• Monitors clusters with Pivotal Command Center. |
| Starting and stopping | • Starts and stops the cluster or individual services.<br>• 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. |

## Deployment Options

The following table illustrates the deployment options and limitations:

| Component | | CLI install | Manual install (rpm) |
|---|---|---|---|
| Command Center (installs the CLI) | | | ✓ |
| Hadoop MR2: HDFS, YARN | | ✓ | ✓ |
| Pig | | ✓ | ✓ |
| Hive | | ✓ | ✓ |
| HBase | | ✓ | ✓ |
| Mahout | | ✓ | ✓ |
| Zookeeper | | ✓ | ✓ |
| Flume | | | ✓ |
| Sqoop | | | ✓ |
| Oozie | | | ✓ |
| Hamster | | | ✓ |
| GraphLab | | | ✓ |
| Advanced Database Services: | HAWQ | ✓ | ✓ |
| | PXF | ✓ | ✓ |

# Planning your Pivotal HD Cluster Deployment

To deploy a Hadoop cluster, Pivotal recommends that you consider the following:

- Select the appropriate hardware configuration for Admin and cluster nodes.
- Map Hadoop services roles to cluster nodes.
- Configure the roles to effectively leverage the underlying hardware platform.

## 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.

⚠ 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

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

## 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.

# Chapter 3 PHD Installation Checklist

This section is intended to assist you in planning and performing your PHD installation. It includes an installation prerequisite checklist and a brief summary of the installation steps. This section is intended for more advanced users; first time users should following the more detailed instructions we provide in Installing PHD Using the CLI.

**Topics:**

## Prerequisites

In addition to specific environment prerequisites, 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

| Step | Task | Description | Completed |
|---|---|---|---|
| 1 | **DNS Lookup** | Verify that hosts can reach each other using hostnames and IP addresses.<br><br>`$ ping -c myhost.mycompany.com // The return code should be 0`<br>`$ ping -c 3 192.168 . 1.2 // The return code should be 0` | |
| 2 | **Check JDK** | Ensure you're running Oracle Java JDK Version 1.7 on the Admin node.<br><br>Java version 1.7 is required; 1.7u15 is recommended.<br><br>As `root`, run:<br><br>`$ /usr/java/default/bin/java -version`<br><br>If not, download and install the appropriate version from Oracle. | |
| 3 | **Package Accessibility** | Verify that all hosts have yum access to an EPEL yum repository.<br>`$ yum list < LIST_OF_PACKAGES >`<br>See Package Accessibility for more details and a list of packages.<br><br>Note that this is not required if the required rpms are accessible locally. | |
| 4 | **Verify iptables is turned off** | (as `root`)<br>`$ chkconfig iptables off`<br>`$ service iptables stop`<br>`$ service iptables status`<br>`iptables: Firewall is not running.` | |
| 5 | **Disable SELinux** | (as `root`)<br>`$ echo 0 > /selinux/enforce` | |

## Installation Steps

The table below briefly describes the steps you need to take to install PHD; more detailed instructions are provided in Installing PHD Using the CLI:

| Step | Task | Details | Completed |
|---|---|---|---|
| 1 | **Install Pivotal Command Center** | `(as root)`<br><br>1. Create a directory (phd) for your PCC installation:<br>`# mkdir phd`<br><br>2. Copy tar file to your specified directory on the admin node, for example:<br>`# scp ./PCC-2.2.x.version.build.os.x86_64.tar.gz host:/root/phd/`<br><br>3. Login as `root` and untar to that directory:<br>`# cd /root/phd`<br>`# tar --no-same-owner -zxvf PCC-2.2.x.version.build.os.x86_64.tar.gz`<br><br>4. Run the installation script from the directory where it was extracted:<br>`# ./install`<br><br>5. As the rest of the installation is done as the `gpadmin` user, change to that user:<br>`# su - gpadmin`<br><br>6. If necessary, enable Secure Connections (see Enable Secure Connections for details) | |
| 2 | **Import JDK**<br><br>See Import JDK | Deploy the downloaded JDK to the cluster nodes<br><br>`$ icm_client import -r <PATH TO JDK>` | |

| Step | Task | Details | Completed |
|---|---|---|---|
| 3 | **Copy the PHD services packages to the Admin node.**<br><br>See Copy the PHD Service Packages for more details. | (as `gpadmin`)<br><br>1. Copy the Pivotal HD services (PHD, ADS (HAWQ)) tarballs from the initial download location to the `gpadmin` home directory ( `home/user/gpadmin`).<br>2. Change the owner of the packages to `gpadmin` then untar the tarballs.<br><br>For example: If the file is a `tar.gz` or `tgz`, use:<br><br>`tar -zxf packagename.tgz`<br><br>If the file is a `tar`, use:<br><br>`tar -xf packagename.tar`<br><br>**Note**: If you want to use GemFire XD, you also need to import and enable the PRTS package. Complete instructions are in the Configuring GemFire XD section. | |
| 4 | **Import the PHD services**<br><br>See Import PHD and Import HAWQ for more details | (as `gpadmin`)<br><br>For each service (PHD, ADS) you are importing, run the following:<br><br>`# icm_client import -s <PATH TO EXTRACTED PHD SERVICE TAR BALL>` | |
| 5 | **Fetch the default Cluster Configuration template** | (as `gpadmin`)<br>`$ icm_client fetch-template -o ~/ClusterConfigDir`<br>**Note**: `ClusterConfigDir` is created automatically. | |
| 6 | **Edit the default Cluster Configuration template** ( `clusterConfig.xml`)<br><br>See Cluster Configuration Files<br>for more details | (as `gpadmin`)<br>At a minimum, you must replace all instances of your selected services with valid hostnames for your deployment.<br><br>**Notes**:<br><br>If you want to use GemFire XD, you need to add that service to the `clusterConfig.xml` file. Complete instructions are available in the Configuring GemFire XD section.<br><br>If you want to enable HA, you need to make some HA-specific changes to some configuration files. Complete instructions are available in the High Availability section. | |
| 7 | **Configure other Pivotal HD and ADS Components** | (as `gpadmin`)<br>Optional: Configure HAWQ and other stack components in their corresponding configuration files (for example: `hawq/gpinitsystem_config` file), as needed | |

| Step | Task | Details | Completed |
|---|---|---|---|
| 8 | **Deploy/Install a cluster**<br><br>See Deploying the Cluster for more details | (as `gpadmin`)<br>`$ icm_client deploy -c ~/ClusterConfigDir`<br><br>**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 that user before running this command. | |
| 9 | **Start the Cluster** | (as `gpadmin`)<br>`$ icm_client start -l < CLUSTER_NAME >` | |
| 11 | **Initialize and Configure HAWQ** | (as `gpadmin`)<br><br>First verify HDFS is running:<br><br>`$ ssh < NAME_NODE >`<br>`$ hdfs dfs -ls /`<br><br>Then, `ssh` to the HAWQ master, the run the following:<br>`$ source /usr/local/hawq/greenplum_path.sh $ /etc/init.d/hawq init`<br><br>If you have a HAWQ standby master configured, initialize that:<br><br>`$ gpinitstandby -s <STANDBY HAWQ MASTER FQDN>` | |

# 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).

A PHD Installation Checklist provides a brief overview of the prerequisites and installation instructions; this section provides more detailed information.

**Topics:**

## PHD Prerequisites

Before you begin your installation; we recommend you have working knowledge of the following:

- **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

Additionally; the following prerequisites are required:

1. **DNS lookup**. 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 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 that you are running Oracle JAVA JDK version 1.7 on the Admin node.

   Version 1.7 is required; version 1.7u15 is recommended.

   As `root`:

```
$ /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:

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

3. **Package Accessibility**. Verify that all packages are available in a local yum repository or that you have yum access to an EPEL yum repository. See Package Accessibility, below.
4. **iptables**. Verify that iptables is turned off:
As `root`:

```
$ chkconfig iptables off
$ service iptables stop
```

5. **SELinux**. Verify that SELinux is disabled:

   As `root`:

   ```
   $ sestatus
   ```

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

   ```
   SELinuxstatus: disabled
   ```

   or

   ```
   SELinux status: permissive
   ```

   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`:

   ```
   $ echo 0 > /selinux/enforce
   ```

   ⚠ 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).

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

   Change the value for the SELINUX parameter to:
   `SELINUX=disabled`

   Then reboot the system.

## Package Accessibility

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 deployed on your cluster. See Creating a YUM EPEL Repository 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

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

```
$ yum list nc postgresql-devel
```

# Installation Instructions

This section provides detailed installation steps. If you are an advanced user you may want to use the more succinct PHD Installation Checklist.

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

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

Also avoid using underscores as they are invalid characters in hostnames.

## Install Pivotal Command Center

1. Download the PCC package from Pivotal Network .
2. As `root`, 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-2.2.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-2.2.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-2.2.x.version`

   For example:

   ```
   $ cd PCC-2.2.x.version
     $ ./install
   ```

   You will see installation progress information on the screen. 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/greenplum-cc/ssl/<servername>.cert`

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

   d. Restart PCC by running:

      ```
      $ service commander restart
      ```

      .

   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
```

## Import 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`

**Notes**

- If you want to use GemFire XD, you also need to copy and import the PRTS package. Complete instructions are in the Configuring GemFire XD section.
- Run `import` each time you wish to sync/import a new version of the package.

## Import JDK

Note that having 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.

1. Download a supported JDK package from
   http://www.oracle.com/technetwork/java/javase/downloads/index.html.
   PHD requires an rpm package, for example: jdk-7u15-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) from Pivotal Network .
2. Copy the Pivotal HD, and optionally ADS (HAWQ) 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:

```
# If the file is a tar.gz or tgz, use
$ tar zxf PHD-2.0.x-<BUILD>.tar.gz

# If the file is a tar, use
$ tar xf PHD-2.0.x-<BUILD>.tar

# Similarly for the Pivotal ADS 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
```

## Import PHD

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

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

Example:

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

## Import HAWQ/PXF

⚠ This is required only if you wish to deploy HAWQ.

1. As `gpadmin`, import the following tar balls for HAWQ and PXF:

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

   For example:

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

# Editing the Cluster Configuration Files

We provide 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.

## 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.

Manually modify this template and use it as input to subsequent commands.

1. 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). This directory contains files that describe the topology of the cluster and the configurations for the various services installed on the cluster.

## 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-1.1.0.0services`: Configure the services to be deployed. By default, every service that Pivotal HD Enterprise 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, data node, 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, `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.

    **Important**

    - `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.
    - We recommend you have multiple disk mount points for datanodes, but it is not a requirement.

    d. **GemFire**. If you want to use GemFire XD, you need to add that service to the `clusterConfig.xml` file. Complete instructions are available in the Configuring GemFire XD section.

    e. **High Availability.** If you want to enable HA, you need to make some HA-specific changes to the `clusterConfig.xml` file and additionally edit some other configuration files. Complete instructions are available in the High Availability section.

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.

⚠ 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.

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

## Configuring HAWQ

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:
  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 0 to 2:
  `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).

## Deploying 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 does three steps:

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.
2. Verifies the prerequisites (internally runs `scanhosts` command)
3. Deploys the cluster

⚠ `scanhosts` and `preparehosts` were commands that in previous releases could be run independently. As of release 2.0.1 that is no longer supported and they are only run internally as part of the `deploy` command.

⚠ Deploying multiple clusters at one time is not supported; deploy one cluster at a time.

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`

**On Cluster Nodes**

- `/tmp/GPHDNodeInstaller_XXX.log`

**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. 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
```

**Your Pivotal HD installation is now complete.**

**You can now start a cluster and start HAWQ.**

## Starting the Cluster

1. As `gpadmin`, start your cluster.

Example:

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

See Managing a Cluster for more detailed instructions and other start up options.

# Initializing HAWQ

1. Verify HDFS is running (you will not be able to initialize HAWQ if HDFS is not running).
   Logon to the client node, name node or data node 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
   ```

2. As `gpadmin`, initialize HAWQ from the HAWQ master.
   Note that HAWQ is implicitly started as part of the initialization.

   `ssh` to the HAWQ Master before you initialize HAWQ
   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
   ```

3. If you have a HAWQ Standby master in your cluster configuration, initialize that by running the following:

   ```
   $ gpinitstandby -s <HAWQ STANDBY MASTER FQDN>
   ```

**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.matastore.uri` in the `hive-site.xml` file on the Namenode to point to that location.

See Managing HAWQ sections for more detailed instructions.

## Verifying 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 Sample Programs 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'
```

# 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. |

# Running Sample Programs

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

## Testing Hadoop

Hadoop commands can be executed from any configured hadoop nodes.
You can run Map reduce 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*
```

When you run a map reduce job as a custom user, *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 HBase

You can test HBase from the HBase master node

```
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
```

## Testing HAWQ

⚠ 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 the 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 the 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 PXF

## Testing PXF on Hive

Make sure you created a 'passwords' table on Hive, which is described in "Testing Hive" section.

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&R
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=HBas
) 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&
FORMAT 'TEXT' (DELIMITER = ',');

select * from ranking order by rank;
```

## Creating a YUM EPEL 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 deployed 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.

# High Availability (HA)

- High availability is disabled by default.
- Currently we only support Quorum Journal based storage for high availability.

To enable HA for a new cluster; follow the instructions below.

To enable HA for an existing cluster, see *Enabling High Availability on a Cluster* in Administering PHD Using the CLI for details.

## HA Best Practices

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, and equivalent hardware to that which would be used in a non-HA cluster.
- 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.

In an HA cluster, the Standby NameNode also performs checkpoints of the namespace state; therefore, it is not necessary to run a Secondary NameNode, CheckpointNode, or BackupNode in an HA cluster. In fact, to do so would be an error. This also allows someone who is reconfiguring a non-HA-enabled HDFS cluster to be HA-enabled to reuse the hardware they had previously dedicated to the Secondary NameNode.

## Setting up a New Cluster with HA

1. Follow the Installation Instructions earlier in this document, then at the point where you are fetching and editing the Cluster Configuration Files, do the following:

   To enable HA, you then must make HA-specific edits to the following configuration files:

   - `clusterConfig.xml`
   - `hdfs/hdfs-site.xml`
   - `hdfs/core-site.xml`
   - `hbase/hbase-site.xml`
   - `yarn/yarn-site.xml`

   ⚠ When specifying the `nameservices` in the `clusterConfig.xml`, do not use underscores ('_'), for example, `phd_cluster.`

2. Edit `clusterConfig.xml` as follows:

   Comment out `secondarynamenode` role in `hdfs` service.

   Uncomment `standbynamenode` and `journalnode` roles in `hdfs` service.

   Uncomment `nameservices`, `namenode1id`, `namenode2id`, `journalpath`, and `journalport` entries in `serviceConfigGlobals.`

3. Edit `hdfs/hdfs-site.xml` as follows:
   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 -->
```

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>
```

4. Edit `yarn/yarn-site.xml`:

```
<property>
    <name>mapreduce.job.hdfs-servers</name>
    <value>hdfs://${nameservices}</value>
</property>
```

5. Edit `hdfs/core-site.xml` as follows:

   Set the following property key 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>
```

   Uncomment following property:

```
<property>
   <name>ha.zookeeper.quorum</name>
   <value>${zookeeper-server}:${zookeeper.client.port}</value>
 </property>
```

6. Edit `hbase/hbase-site.xml` as follows:
   Set the following property key 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>
```

7. 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
```

8. 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>
```

9. If the cluster is not already deployed, continue configuring your cluster as described earlier in this document; then deploy (see Deploying the Cluster).

⚠ If you are using an initialized version of HAWQ and need to use the HA feature, see the Pivotal HAWQ Administration Guide for more information.

# Configuring GemFire XD

Pivotal HD Enterprise 2.x provides support for GemFire XD 1.0. GemFire XD is optional and is distributed separately from other PHD components.

GemFire XD is installed via the CLI. CLI installation instructions and configuration steps are provided below. GemFire XD can be added during initial deployment, like any other service, or can be added during a reconfiguration of a cluster.

Further operational instructions for GemFire XD are provided in the *Pivotal GemFire XD User's Guide.*

## Overview

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

## 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 |

## Best Practices

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

## Enabling PRTS Services

Follow the instructions below to add GemFire XD before you deploy or reconfigure a cluster.

If you wish to deploy Gemfire XD Beta, perform the following:

1. Download the PRTS tarball from the initial download location to the gpadmin home directory.

2. Change ownership of the packages to gpadmin and untar. For example:
   If the file is a `tar.gz` or `tgz`:`tar zxf PRTS-1.0.x-<BUILD>.tgz`
   If the file is a `tar`:`tar xf PRTS-1.0.x-<BUILD>.tar`

3. As `gpadmin`, enable the PRTS service:

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

```
$ icm_client import -s PRTS-1.0.x-<BUILD>/
```

4. Edit the Cluster Configuration file as follows:

   **During initial deployment**: Retrieve the `clusterConfig.xml` file using the `icm_client fetch-template` command. See Cluster Configuration Files for more details.

   **Adding to an exiting cluster**: Edit the `clusterConfig.xml` file (`icm_client fetch-configuration`), then reconfigure the cluster (`icm_client reconfigure`). See Reconfiguring a Cluster.

   - Open `clusterConfig.xml` and 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>`

## GemFire XD Notes

**NOTE 1:**
Gemfire XD binaries have been deployed at this point, but each node is not configured until needed.

```
<gfxd>
<gfxd-locator>host1</gfxd-locator>
<gfxd-server>host2</gfxd-server>
</gfxd>
# but host1 does not act as server upon service gfxd start command at this point.
```

Refer to the *Pivotal GemFire XD User's Guide* to complete the configuration.

**NOTE 2:**

You cannot start GemFire XD (gfxd) using the `icm_client start` command. Refer to the *Pivotal GemFire XD User's Guide* for instructions about starting your gfxd services.

## Managing GemFire XD

Refer to the *Pivotal GemFire XD User's Guide*.

A *Quick Start Guide* that includes instructions for starting and stopping gfxd servers and locators is also available, here:
http://gemfirexd.docs.gopivotal.com/latest/userguide/index.html?q=getting_started/15-minutes.html

# Installing 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 |
| **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` 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 5 Upgrade Checklist

This section is intended to assist you in planning and performing your PHD upgrade. It includes a upgrade prerequisite checklist and a brief summary of the upgrade steps. This section is intended for more advanced users; first time users should following the more detailed instructions we provide in Upgrading PHD Using the CLI.

You can find more detailed instructions for upgrading your system in Upgrading PHD Using the CLI.

# Before You Begin

## Prerequisites

| Step | Task | Description | Completed |
|---|---|---|---|
| 1 | **PADS file location** | If you are upgrading PADS, make note of the path to the extracted pre-upgrade PADS tar ball. If you don't remember, you can just download it again and untar it. | |
| 2 | **Backup Data** | We recommend that you backup any critical data before running any upgrade. | |
| 3 | **Backup Service Configuration File(s)** | Backup the configuration files of any services you will be manually reconfiguring, post CLI-upgrade. | |
| 4 | **JDK 1.7** | Make sure you are running JDK 1.7. If you are not, download it from Oracle.<br><br>⚠ That this is a new requirement; prior to PHD 2.0, JDK 1.6 was also supported. | |
| 5 | **Compact HBase Tables** | Hbase 0.96 only supports HFileV2 and compacting tables rewrites HFileV1 format to HFileV2**.** | |
| 6 | **GemFireXD** | The PHD 2.0 upgrade does not support and upgrade of the GemFireXD service.<br><br>You will have to remove the GemFireXD beta service prior to PHD upgrade; followed by a fresh install of GemFireXD. Data migration from GemfireXD beta is not supported. | |

## Upgrade Steps

The table below briefly describes the steps you need to take to upgrade a cluster; more detailed instructions are provided in Upgrading PHD Using the CLI:

| Step | Task | Details | Completed |
|---|---|---|---|
| 1 | Verify the state of your cluster | Make sure your cluster is healthy and in a consistent state.<br><br>Run `hdfs dfsadmin -report`<br><br>Run `fsck` | |
| 2 | Back up Hive metastore | Login to the machine running the hive metastore database, then run:<br><br>`pg_dump -U hive -p 10432 metastore > hive_metastore_1.backup` | |

| Step | Task | Details | Completed |
|---|---|---|---|
| 3 | Revert to Non-High Availability | If High Availability is enabled, disable it before you begin your upgrade. See Disabling High Availability on a Cluster for instructions.<br><br>To complete this step, run the following SQL command:<br><br>`psql -U postgres -p 10432 gphdmgr -c "UPDATE cluster_properties SET property_value='false' WHERE <cluster_id>=2 AND property_name='cluster.nn.isHAEnabled';"` | |
| 4 | Revert to Non-Secure | If security is enabled, disable it before you begin your upgrade. See Disabling Security on a Cluster for instructions. | |
| 5 | Remove Standby HAWQ master | Remove Standby HAWQ master:<br><br>Source the `greenplum_path.sh`:<br><br>`$ source /usr/local/hawq/greenplum.path.sh`<br><br>Then, as `gpadmin`, run:<br><br>`$ gpinitstandby -r` | |
| 6 | Stop Services | Stop HAWQ (if applicable):<br><br>`$ /etc/init.d/hawq stop`<br><br>(See Managing HAWQ for details.)<br><br>As `gpadmin`, stop all PHD services :<br><br>`$ icm_client stop -l <CLUSTER NAME>`<br><br>(See Managing a Cluster for details.)<br><br>As `root`, stop PCC:<br><br>`$ service commander stop` | |
| 7 | Import and Upgrade PCC | Untar the new PCC package, then run (as `root`):<br><br>`$ ./install`<br><br>Change the user to `gpadmin` for the rest of the upgrade | |
| 8 | CLI Self Upgrade | `$ icm_client self-upgrade` | |
| 9 | Import HAWQ (PADS) | Run:<br><br>`$ icm_client import -s < PATH TO EXTRACTED HAWQ TAR BALL >` | |

| Step | Task | Details | Completed |
|---|---|---|---|
| 10 | Upgrade HAWQ (PADS) | To upgrade HAWQ run:<br>`$ icm_client upgrade -l <CLUSTERNAME> -s pads -o < PATH TO EXTRACTED OLD ADS TAR BALL > -n < PATH TO EXTRACTED NEW ADS TAR BALL >`<br>Then, to migrate data, on the HAWQ master node, run:<br>`gpmigrator <old_HAWQHOME_path> <new_HAWQHOME_path>` | |
| 11 | Import PHD | Run:<br>`$ icm_client import -s < PATH TO EXTRACTED PHD TAR BALL >` | |
| 12 | Upgrade PHD | PHD 2.0.1 requires Oracle JDK 1.7. Get the JDK rpm (for example: `jdk-7u15-linux-x64.rpm`) and include it in the upgrade command as shown below, so that the `upgrade` command can deploy it to the cluster nodes.<br>`$ icm_client upgrade -l <CLUSTERNAME> -s phd -j ~/jdk-7u15-linux-x64.rpm` | |
| 13 | Upgrade Configuration Files | 1. Synchronize configuration files<br>2. Reconfigure the cluster<br>⚠ Do not add any security or HA-specific configuration parameters/values at this time, wait until you have completed the upgrade. | |
| 14 | Upgrade HDFS | • Backup Name Node data<br>• Run `HdfsUpgrader.py` with appropriate options (see `Upgrade HDFS` for details) | |
| 15 | Restart Cluster | `$ icm_client restart -l <CLUSTER_NAME>` | |
| 16 | Post-Upgrade HAWQ | Check to see if HAWQ is running, if not, start it now:<br>`$ /etc/init.d/hawq start`<br>Reinitialize HAWQ Standby Master:<br>`$ gpinitstandby -s <standby_hostname>` | |
| 17 | Finalize HDFS Upgrade | Run `FinalizeHDFS` command | |
| 18 | Finalize HBase Upgrade | • Check for HFileV1 data (not supported after upgrade)<br>• Run HBase upgrade | |

| Step | Task | Details | Completed |
|---|---|---|---|
| 19 | Reconfigure Manually Installed Services | 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. | |
| 20 | Re-enable High Availability | See Enabling High Availability on a Cluster for details. | |
| 21 | Re-secure Cluster | We provide instructions for manually enabling Kerberos authentication in the *PHD 2.0 Stack and Tools Reference Guide*. We also can provide scripts to automate this process. To obtain these scripts and instructions how to use them, contact either your PHD Account Manager, or open up a service request with support at https://support.emc.com/ and ask for the PHD Secure Install Tools. | |
| 22 | Move HAWQ Filespace | For HA clusters: For HAWQ, you need to move the HAWQ filespace to HA-enabled HDFS, as described in Moving HAWQ Filespace to HA-enabled HDFS. | |

# Chapter 6 Upgrading PHD Using the CLI

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

See the Upgrade Checklist for a quick summary of the prerequisites and installation steps.

**Topics:**

## Prerequisites

- **PADS file location:** Make note of the path to the extracted pre-upgrade PADS tar ball. If you don't remember, you can just download it again and untar it.
- **Backup Data:** We recommend you backup any critical data before performing any upgrades.
- **Backup 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. Backup the configuration files for these services. See the *Pivotal HD Enterprise Stack Tool and Reference Guide* for the locations of these configuration files.
- **Oracle JDK 1.7**. Ensure that you are running Oracle JAVA JDK version 1.7.0_xx (minimum 1.7.0.15) as the default JDK on the Admin node.

⚠ 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)
```

- **Compact HBase Tables**
  Compact all tables on the existing HBase 0.94 cluster**:** For example: to compact table `t1`, login to the HBase shell, then run:
  `major_compact 't1'`

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

- **Remove GemFireXD**

The PHD 2.0 upgrade does not support an upgrade of the GemFireXD beta service.

You will have to remove the GemFireXD beta service prior to PHD upgrade; followed by a fresh install of GemFireXD. Data migration from GemFireXD beta is not supported.

To remove GemFireXD, run the following:

```
vim icmconf/clusterConfig.xml # Remove gfxd from <services>
icm_client reconfigure -c icmconf -l test
```

Note that you will see the following error if you attempt to upgrade a cluster with GemFireXD installed.

**Gemfire Upgrade Error Example**

```
-bash-4.1$ icm_client upgrade -l test -s phd
Please ensure you've backed up manually installed service configurations (not installed by
icm_client) if any. Do you wish to proceed with the PHD upgrade ? . (Press 'y' to continue, any
other key to quit): y
Please enter the root password for the cluster nodes:
PCC creates a gpadmin user on the newly added cluster nodes (if any). Please enter a non-empty
password to be used for the gpadmin user:
Starting upgrade

Return Code : 6000
Message : Upgrade Cluster Error
Details :
Cluster Hosts :
        Operation Code : UPGRADE_FAILURE
        Operation Error : GEMFIRE XD must not be present if upgrading
        Log File : /var/log/gphd/gphdmgr/gphdmgr-webservices.log

[==========================================================================================
100%
Results:
centos64-5... [Success]
centos64-4... [Success]
centos64-3... [Success]
centos64-2... [Success]
Details at /var/log/gphd/gphdmgr/gphdmgr-webservices.log
[ERROR] Cluster upgrade failed
-
```

Once you've completed your PHD Upgrade, reinstall GemFireXD as a fresh install. See Configuring GemFireXD for details.

## Upgrade Instructions

Follow the instructions below to upgrade your PHD system.

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`. Make sure there are no 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.

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

   **dfsadmin report example**

```
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
```

   **fsck example**

```
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
```

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

2. **Backup 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`, login to the machine running the hive metastore database

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

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

3. **Revert to Non-HA (if applicable)**:
   You cannot upgrade a 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, 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';"
   ```

   Where: `<cluster_id>` is the id of your cluster.
   Note that this SQL command is only necessary for upgrades from 1.1.1 to PHD 2.0.1.

4. **Revert to Non-Secure (if applicable):**
   You cannot upgrade a cluster with security enabled. Revert your cluster to non-secure before proceeding with an upgrade.
   See Disabling Security for details.

5. **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, refer to the *HAWQ Installation and Upgrade Guide*.

6. **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 <CLUSTER NAME>`

   c. As `root`, stop PCC:
      `$ service commander stop`

7. **Import and upgrade PCC:**

   a. Download the new PCC file from Pivotal Network .

   b. Copy the new PCC tar file to your installation directory on the admin node, for example:
   ```
   $ scp ./PCC-2.2.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-2.2.x. version.build.os .x86_64.tar.gz
   ```

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

   ⚠ 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.

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

8. **CLI Self-Upgrade:**
   As `gpadmin`, run the following command to upgrade the CLI:
   ```
   $ icm_client self-upgrade
   ```
   Note that this command may return very quickly. This does not indicate any problems and you can continue with the upgrade.

9. **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 TAR BALL >
   ```

10. **Upgrade HAWQ:**

    ⚠ 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 TAR BALL > -n < PATH TO EXTRACTED NEW ADS TAR BALL >`

    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-1.1.3.0/ /usr/local/hawq-1.2.0.0/ -d /data1/master/gpseg-1
```

       ⚠ If you encounter errors migrating HAWQ data, refer to the *HAWQ Administrator Guide* for help.

    c. Optional: You can delete the old HAWQ rpm file by running:
       `$ yum erase <HAWQ_OLD_RPM_NAME>`

11. **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 TAR BALL >`

12. **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.
    Only clusters running the following versions can be upgraded to use the PHD 2.0.x stack:

    PHD 1.1.1 and PHD 1.1
    PCC 2.1.1 and PCC 2.1

    PHD 2.0.1 requires Oracle JDK 1.7.

    If you are already running JDK 1.7, to upgrade PHD, as `gpadmin`, run:

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

    If you need to upgrade to JDK 1.7, include the JDK rpm in the upgrade command (for example: `jdk-7u15-linux-x64.rpm`) so that the `upgrade` command can deploy it to the cluster nodes:

    ```
    $ icm_client upgrade -l <CLUSTERNAME> -s phd -j ~/jdk-7u15-linux-x64.rpm
    ```
    This upgrades the PHD stack on all cluster nodes.

    Note that all upgrade steps, including post-upgrade configuration steps described below, should be completed before you re-enable HA or security on a cluster.

13. **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 (step c, above) of merging the existing PHD configuration with the `newTemplate`, follow these steps,1. Overwrite `clusterConfig.xml` in `newTemplate` from the one from `origConfiguration` directory: `$> cp ~origConfiguration/clusterConfig.xml ~newTemplate/ClusterConfig.xml`2. Change the value of <`gphdStackVer`> to PHD-2.0.1.0 in the `~newTemplate/clusterConfig.xml`3. If you have explicitly modified any properties from PHD services configuration files, for example, `hdfs/hdfs-site.xml`, `yarn/yarn-site.xml` etc., then make the corresponding changes to these configuration files under `~newTemplate/` directory.

d. Upgrade service by specifying the cluster configuration directory as `~/newTemplate` with your updated contents:

```
icm_client reconfigure -c ~/newTemplate -l <CLUSTERNAME>
```

14. **Upgrade HDFS:**

    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 Nameode name directories. The location of the Namenode name directories is specified in the value of `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 Namenode metadata by running:

    ```
    /usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py -l <CLUSTER NAME> -o
    backupNNMetadata -s 2.0.5_alpha_gphd_2_1_1_0 -t 2.2.0_gphd_3_0_0_0

    # Source prefix would be 2.0.5_alpha_gphd_2_1_0_0 instead of 2.0.5_alpha_gphd_2_1_1_0 if you
    are upgrading from (PHD-1.1.0.0)
    ```

    b. Run NameNode upgrade by running:

    ```
    /usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py -l <CLUSTER NAME> -o
    nnupgrade -s 2.0.5_alpha_gphd_2_1_1_0 -t 2.2.0_gphd_3_0_0_0
    # Source prefix would be 2.0.5_alpha_gphd_2_1_0_0 instead of 2.0.5_alpha_gphd_2_1_1_0 if you
    are upgrading from (PHD-1.1.0.0)
    ```

    c. Run Data Node upgrade by running:

    ```
    /usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py -l <CLUSTER NAME> -o
    dnupgrade -s 2.0.5_alpha_gphd_2_1_1_0 -t 2.2.0_gphd_3_0_0_0
    # Source prefix would be 2.0.5_alpha_gphd_2_1_0_0 instead of 2.0.5_alpha_gphd_2_1_1_0 if you
    are upgrading from (PHD-1.1.0.0)
    ```

15. **Restart the cluster:**

    `As :$ icm_client restart -l <CLUSTER_NAME>`

16. **Post-Upgrade HAWQ:**

    a. On the HAWQ master, as `gpadmin`:
    Check HAWQ status:
    `$ /etc/init.d/hawq status`
    If it is not running, start it by running: `$ /etc/init.d/hawq start`

    b. 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, refer to the *HAWQ Installation and Upgrade Guide*.

17. **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 upgrade process:

```
/usr/bin/python /usr/lib/gphd/gphdmgr/lib/client/HdfsUpgrader.py -l <CLUSTER NAME> -o
finalizeUpgrade -s 2.0.5_alpha_gphd_2_1_1_0 -t 2.2.0_gphd_3_0_0_0
```

HBase master will not start unless the HBase upgrade is finalized. Please ensure HDFS upgrade is finalized before finalizing HBase upgrade.

18. **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.

    ⚠ As part of the prerequisites you should have already compacted all the tabels on the existing HBase cluster; this will have overwritten any HFileV1 data to HFileV2 format.

    b. Make sure Zookeeper and HDFS are running but HBase is stopped, then run:
    ```
    $ sudo -u hbase hbase upgrade -execute
    ```
19. **Reconfigure Manually Installed Services:**
    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. Refer to the *Pivotal HD Enterprise Stack and Tool Reference Guide* for details.

    ⚠ Backing up the configuration files for these services is a prerequisite for this upgrade procedure. See the *Pivotal HD Enterprise Stack Tool and Reference Guide* for the locations of these configuration files.

20. **Re-enable HA:**
    See Enabling High Availability on a Cluster for details.
21. **Re-Secure:**
    We provide instructions for manually enabling Kerberos authentication in the *PHD 2.0 Stack and Tools Reference Guide*. We also can provide scripts to automate this process. To obtain these scripts and instructions how to use them, contact either your PHD Account Manager, or open up a service request with support at https://support.emc.com/ and ask for the PHD Secure Install Tools.
22. **For HA Clusters: Move HAWQ filespace to HA enabled HDFS:**
    For HAWQ, you need to move the HAWQ filespace to HA-enabled HDFS, as described below.

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. Installing PHD Using the CLI includes a section *Running Sample Programs* that provides instructions for testing the various services.

# Moving HAWQ Filespace to HA-enabled HDFS

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. 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, you can do this by moving the filespace from hdfs://phdcluster/hawq-security to hdfs://phdcluster/hawq/another/path.

## Stopping HAWQ Cluster and Backup 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 backup the catalog. This is to ensure that you do not lose data due to a hardware failure or during an operation (such as killing HAWQ process).

1. Disconnect all workload connections.
2. Issue a checkpoint.
3. Shutdown the HAWQ cluster.
4. Copy the master data directory:

   ```
   $MASTER_DATA_DIRECTORY /catalog/backup/location
   ```

## Moving 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 file space location:

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

**Notes**

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

**Important**

Errors while moving the location of the filespace:

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 teh 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 configured standby master you can skip this task.

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

# Upgrade Reference Information

## 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 HADOOP-8886 |
| mapred.outdir.resolverClass | org.apache.hadoop.mapreduce.DefaultPathResolver | |
| kfs.client-write-packet-size | 65536 | KFS is no longer supported, see HADOOP-8886 |
| kfs.blocksize | 67108864 | KFS is no longer supported, see HADOOP-8886 |
| kfs.bytes-per-checksum | 512 | KFS is no longer supported, see HADOOP-8886 |
| kfs.replication | 3 | KFS is no longer supported, see 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 |
| 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/rmst |
| yarn.nodemanager.resource.cpu-cores | yarn.nodemanager.resource.cpu-vcores | 8, See YARN-782 |
| yarn.nodemanager.aux-services. mapreduce.shuffle.class | yarn.nodemanager.aux-services. mapreduce_shuffle.class | org.apache.hadoop.mapred.Shufflel |
| 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.<br>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.<br>Will be added when the patch is added again to hadoop 2.2.0 |
| yarn.dynamic.resource.vcores.maximum | -1 | Introduced by hve patch.<br>Will be added when the patch is added again to hadoop 2.2.0 |
| yarn.dynamic.resource.enable | true | Introduced by hve patch.<br>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.<br>Will be added when the patch is added again to hadoop 2.2.0 |
| yarn.dynamic.resource.vcores.minimum | 0 | Introduced by hve patch.<br>Will be added when the patch is added again to hadoop 2.2.0 |
| yarn.nodemanager.vcores-pcores-ratio | 2 | See 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.capacity.ProportionalCapacityPreemptionPolicy |
| 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:

| Name | Old Default Value | New Default Value |
|---|---|---|
| dfs.namenode.checkpoint.txns | 40000 | 1000000 |
| dfs.blocksize | 67108864 | 134217728 |

### *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 |

| Name | Default Value |
|---|---|
| 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 |
| mapreduce.jobhistory.cleaner.interval-ms | 86400000 |
| mapreduce.jobhistory.client.thread-count | 10 |
| mapreduce.jobhistory.move.interval-ms | 180000 |

| Name | Default Value |
|---|---|
| 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 |

| Name | Old Default Value | New Default Value |
|---|---|---|
| hbase.regionserver.hlog.reader.impl | org.apache.hadoop.hbase.regionserver.wal.SequenceFileLogReader | org.apache.hadoop.hbase.regio wal.ProtobufLogReader |
| hbase.regionserver.hlog.writer.impl | org.apache.hadoop.hbase.regionserver.wal.SequenceFileLogWriter | org.apache.hadoop.hbase.regio wal.ProtobufLogWriter |
| 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.WritableRpcEngine -> org.apache.hadoop.hbase.ipc.ProtobufRpcServerEngine |
| io.storefile.bloom.cacheonwrite | hfile.block.bloom.cacheonwrite | false (See 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 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 |

| Name | Default Value |
|---|---|
| hbase.thrift.htablepool.size.max | 1000 |
| hbase.hstore.bytes.per.checksum | 16384 |
| hbase.config.read.zookeeper.config | false |
| hbase.master.loadbalancer.class | org.apache.hadoop.hbase.master. balancer.StochasticLoadBalancer |
| 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

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 7 Administering PHD Using the CLI

This section describes the administrative actions that can be performed via Pivotal Command Center's command line interface (CLI).

**Topics:**

# Managing a Cluster

## Starting a Cluster

You can use the `start` command to start all the configured services of the cluster, to start individual services configured for the cluster, and to start individual roles on a specific set of hosts.

```
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
```

The following table describes the list of values for the HDFS, MapRed, ZooKeeper, HBase, and HAWQ services:

| Option | Description |
|---|---|
| `start` | Starts all configured cluster services in the right topological order based on service dependencies. |
| `-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. |

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.

**Notes**

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.

For example:
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 `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.

```
[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
```

The following table describes the list of values for the HDFS, MapRed, ZooKeeper, HBase, and HAWQ services.

| Option | Description |
|---|---|
| `stop` | Stops all configured cluster services in the right topological order based on service dependencies. |
| `-s` | Stops the specified service and all the dependent services in the right topological order. The supported services are hdfs, yarn, zookeeper, hbase, hive, hawq, pig, and mahout. |
| `-r` | Stops the specified role on a specific set of hosts. Hosts can be specified using the -o option. |

For example:
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 `-restart` command to stop, then restart a cluster.

See stopping and starting a cluster, above, for more details about the stop/start operations.

```
[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 `reconfigure` command to update specific configuration for an existing cluster.

**Caution**

Running the `reconfigure` command on a secure cluster will disable security.

Some cluster specific configurations cannot be updated:

**Important**

- Reconfiguring the topology of a cluster (host to role mapping) is not allowed. For example: changing the NameNode to a different node or adding 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 a `--force` option.

- You are expected to take care of all the necessary prerequisites prior to making changes to any of the following properties using the force flag
- Incorrect provisioning can make the cluster get into an inconsistent/unusable state

| 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` |

```
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)
```

**To reconfigure an existing cluster:**

1. Stop the cluster:
   `icm_client stop -l CLUSTERNAME`

2. Fetch the configurations for the cluster in 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`

Following an upgrade or reconfiguration, you need to synchronize the configuration files, as follows:

1. Fetch the new templates that come with the upgraded software by running `icm_client fetch-template`.

2. Retrieve the existing configuration from 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 service by specifying the cluster configuration directory with updated contents.

## Add / Remove Services

Services can be added / removed using `icm_client reconfigure` command.

- Edit the `clusterConfig.xml` file to add or remove services from the service list in `services` tag
- Edit `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.
- As 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

⚠ 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

## Add Hosts to Cluster

If you plan to add hosts as part of adding a new service, perform the following:

- Prepare the new hosts using the `icm_client preparehosts` command
- Refer to *Add / Remove Services* section

If you plan to add/remove hosts as part of an existing service in the cluster do the following:

⚠ You can only add or remove hosts for slave roles (refer to *Expanding a Cluster* section 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

⚠ 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 Configuration about a Deployed Cluster

Run the `fetch-configuration` command to fetch the configurations for an existing cluster and store them in a local file system directory.

```
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
```

**Sample Usage**

```
icm_client fetch-configuration -l CLUSTERNAME -o LOCALDIR
```

## Listing Clusters

Run the `list` command to see a list of all the installed clusters:

```
[gpadmin]# icm_client list --help
Usage: icm_client list [options]

Options:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
```

**Sample Usage**:

```
icm_client list
```

## Expanding a Cluster

**Notes**

- Make sure you run `preparehosts` against the new slave hosts prior to adding them to the cluster. (See the `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 `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 `reconfigure` command (see Reconfiguring a Cluster).

```
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
                      addded slave nodes. Default:
                      /usr/lib/gphd/gphdmgr/hawq_sys_config/. Ignored if -p
                      is specified
```

**Sample Usage:**

```
icm_client add-slaves -l CLUSTERNAME -f slave_hostfile
```

Make sure you start 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.

If HAWQ is configured, refer to the *Expanding HAWQ* section

Hive does not have any slave roles, and therefore cannot be provisioned for an expansion.

## Shrinking a Cluster

Make sure you decommission the slave hosts (refer to the next section) prior to removing them to avoid potential data loss.

Run the `remove-slaves` command lets the user to remove slave hosts from an existing cluster. All the slave roles for the existing cluster services will be removed from the given hosts.

```
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.
```

**Sample Usage**:

```
icm_client remove-slaves -l CLUSTERNAME -f hostfile
```

## Decommissioning Nodes

Decommissioning is required to prevent potential loss of data blocks when you shutdown/remove slave hosts form a cluster. This is not an instant process since it requires replication of potentially a large number of blocks to other cluster nodes.

The following are the manual steps to decommission slave hosts (datanodes,nodemanagers) from a cluster.

- On the NameNode host machine
  - Edit the `/etc/gphd/hadoop/conf/dfs.exclude` file and add the datanode hostnames to be removed (separated by newline character). Make sure you use the FQDN for each hostname.
  - Execute the dfs refresh command

    ```
    [gpadmin] sudo -u hdfs hdfs dfsadmin –refreshNodes
    ```

- On the Yarn Resource Manager host machine
  - Edit `/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.
  - Execute the yarn refresh command

    ```
    [gpadmin] sudo -u hdfs yarn rmadmin -refreshNodes
    ```

- Check Decommission status
  - Monitor decommission progress with name-node Web UI `http://NAMENODE_FQDN:50070` and navigate to Decommissioning Nodes page
  - Check whether the admin state has changed to Decommission In Progress for the DataNodes being decommissioned. When all the DataNodes report their state as Decommissioned then all the blocks have been replicated.

- Shut down the decommissioned nodes
    - Stop datanode and yarn node manager on the targeted slaves to be removed

```
[gpadmin] icm_client stop -l CLUSTERNAME -r datanode -o hostfile
[gpadmin] icm_client stop -l CLUSTERNAME -r yarn-nodemanager -o hostfile
```

⚠ For HBase regionservers you can proceed with shutting down the region servers on the slave hosts to be removed. It is preferable to use `graceful_stop` script that hbase provides if load balancer is disabled.

## High Availability

### Enabling High Availability on a Cluster

- High availability is disabled by default.
- Currently we only support Quorum Journal based storage for high availability.
- PCC 2.1 is the first version to support HA. If you are running an earlier version, download and import the latest version of Pivotal Command Center (PCC) (see Installing PHD Using the CLI for details)

⚠ Before you enable HA for any cluster, make sure you take into consideration our recommended HA Best Practices.

To enable HA for a new cluster; follow the instructions provided in the *High Availability* section of Installing PHD Using the CLI.

To enable HA for an existing cluster, see below.

1. Stop the cluster:

    ```
    icm_client stop -l CLUSTERNAME
    ```

2. For HAWQ users, stop HAWQ.
   From the HAWQ master, as `gpadmin`, run the following:

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

3. Backup 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:

   Comment out `secondarynamenode` role in `hdfs` service

   Uncomment `standbynamenode` and `journalnode` roles in `hdfs` service

   Uncomment `nameservices`, `namenode1id`, `namenode2id`, `journalpath`, and `journalport` entries in `serviceConfigGlobals`

6. Edit `hdfs/hdfs-site.xml` as follows:
   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 -->
```

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. Edit `yarn/yarn-site.xml` as follows:  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:

   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>
```

Then uncomment following property:

```
<property>
   <name>ha.zookeeper.quorum</name>
   <value>${zookeeper-server}:${zookeeper.client.port}</value>
 </property>
```

9. Edit `hbase/hbase-site.xml` as follows:
   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
```

14. Start the cluster:

```
icm_client start -l CLUSTERNAME
```

**Caution**

Running the `reconfigure` command on a secure cluster disables security in PHD-1.1.0.0 and PHD-1.1.1.0

15. Update the HIVE Metastore:
Hive metastore contains references to `hdfs` path with `namenode:port` in the url. This needs to be updated to use the nameservices so HIVE scripts can work when ever NameNode failure happens.
**Note**: Make sure metastore is not running and is backed up to a persistent store before running the update commands.

    a. Login to host configured as `hive-metastore`.

b. Display the current NameNode and hdfspath for 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 and `current_namenode` is the hostname of the NameNode on the cluster before reconfiguring to enable HA.

⚠ When specifying the `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
```

## Disabling High Availability on a Cluster

**Important**

You must disable high availability before upgrading a cluster.

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:

   Uncomment out `secondarynamenode` role in `hdfs` service

   Comment `standbynamenode` and `journalnode` roles in `hdfs` service

   Comment `nameservices`, `namenode1id`, `namenode2id`, `journalpath`, and `journalport` entries in `serviceConfigGlobals`

7. Edit `hdfs/hdfs-site.xml` as follows:
   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 -->
```

Uncomment 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`

```
<property>
    <name>mapreduce.job.hdfs-servers</name>
    <value>hdfs://${namenode}:${dfs.port}</value>
</property>
```

9. Edit `hdfs/core-site.xml` as follows:

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>
```

Comment following property:

```
<property>
   <name>ha.zookeeper.quorum</name>
   <value>${zookeeper-server}:${zookeeper.client.port}</value>
 </property>
```

10. Edit `hbase/hbase-site.xml` as follows:
    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 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>
```

13. Run the following command to reconfigure the cluster with your new configuration file:

```
icm_client reconfigure -l CLUSTERNAME -c LOCALDIR
```

14. Start the cluster:

```
icm_client start -l CLUSTERNAME
```

15. Update the HIVE Metastore:
    Hive metastore contains references to `hdfs` path with `nameservices` in the url. This needs to be updated to use the namenode:port.
    **Note**: Make sure metastore is not running and is backed up to a persistent store before running the update commands.

    a. Login to host configured as `hive-metastore`.

    b. Display the current NameNode and hdfspath for hive warehouse directory:
    `/usr/lib/gphd/hive/bin/metatool -listFSRoot`

    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 and `current_namenode` is the hostname of the NameNode on the cluster after reconfiguring to disable HA.

⚠ When specifying the `nameservices`, do not use underscores ('_'), for example, `phd_cluster`.

16. 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
```

## HAAdmin Command Reference

- `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 state of a given NameNode:

`hdfs haadmin -getServiceState <serviceid> Ex : 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 NameNode:

`hdfs haadmin --failover <serviceid> <serviceid>`

For example:

`hdfs haadmin --failover nn1 nn2`

# Security

PHD clusters can configured to use Kerberos authentication.

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

## Enabling Kerberos Authentication

You must install and configure Kerberos to enable security in Pivotal HD 1.1.x. and higher. Complete instructions are provided in the *PHD Stack and Tools Reference Guide*.

## Disabling Kerberos Authentication

You need to disable Security before upgrading the cluster. To disable security do the following:

1. Stop the cluster

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

2. If you have HBase installed and HBase-to-Zookeeper communication is secured (true in most cases), do the following steps.Tables created while HBase is secure have ACLs set on them which only allow SASL authenticated users to modify them. In order to operate in non-secure mode, you must do the following. You can skip these steps if you don't have HBase installed.

   a. Start just the Zookeeper service.

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

   b. On HBase master:

      i. Run Zookeeper CLI:

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

      ii. Check if there are any regions in transition. Output `[]` means there are NO regions in tranistion 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(CONNECTED) 0]
         ls /hbase/region-in-transition
         []
         ```

         If there are regions in transition, either wait for them to finish (start cluster again) or set ACL to make then 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(CONNECTED) 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(CONNECTED) 2]
ls /hbase/unassigned
[123456789]
[zk: node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTED) 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`** else they won't be readable while security is disabled.

If you're only disabling security temporarily for upgrade and intend to enable it again after upgrade, you may skip setting ACLs on tables.

```
[zk: node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTED) 4]
ls /hbase/table
[hbase:meta, hbase:namespace, testtable]
[zk: node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTED) 5]
getAcl /hbase/table/hbase:meta
'world,'anyone
:cdrwa
[zk: node2.phddev.local:2181,node1.phddev.local:2181,node3.phddev.local:2181(CONNECTED) 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(CONNECTED) 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(CONNECTED) 8]
getAcl /hbase/table/testtable
'world,'anyone
:cdrwa
```

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

**You must 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, for example`/hbase/table`, run the following

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

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

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

c. Stop the Zookeeper service from 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. Make sure it runs successfully on all nodes before proceeding further. Perform the following steps on ICM Admin node.

   a. Fetch the current configuration in a directory `SecureConfiguration`

   ```
   [gpadmin]# icm_client fetch-configuration -o SecureConfiguration
   ```

   b. Copy `SecureConfiguration` to `NonSecureConfiguration`

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

   ⚠ In general, while removing properties you may ignore and proceed further if it's already missing as it may happen depending on how the cluster was secured originally. Similarly, while editing properties if it already has recommended value, you may safely proceed further.

   a. Remove the following 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>
```

b. Remove the following 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>
```

c. Edit the following properties in `hdfs/hdfs-site.xml` to values as 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>
```

d. 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>
```

e. Remove the following from `yarn/yarn-site.xml`. (If present. Please 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>
```

f. Remove the following 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>
```

g. Edit `yarn/container-executor.cfg`.

**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
```

h. Remove the following 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
```

i. Remove the following from `zookeeper/zoo.cfg`

**zookeeper/zoo.cfg**

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

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

j. 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"
```

k. Remove the following 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>
```

l. Remove the following 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"
```

m. Remove the following 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>
```

n. If present, remove the following from `hawq/hdfs-client.xml`

**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>
```

o. Remove the following from `hawq/gpinitsystem_config`

**hawq/gpinitsystem_config**

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

4. Run ICM `reconfigure` using `NonSecureConfiguration` we just modifed to push these changes to cluster nodes:

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

- With Cluster services still stopped, **co mment** the following lines (if present; ignore otherwise) 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=${HADOOP_LOG_DIR}/hdfs
export HADOOP_PID_DIR=/var/run/gphd/hadoop-hdfs/
export HADOOP_SECURE_DN_PID_DIR=${HADOOP_PID_DIR}
```

- 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.

- Remove security from any manually installed service following the reverse of instructions to enable them.
- Start the Cluster. At this point, security should be disabled and you may run test commands to validate data is still accessible in non-secure mode.

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

## Uninstalling a Cluster

You must run the stop command to stop running clusters before running the `uninstall` command. You must also ensure that HAWQ has been stopped before uninstall.

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.

⚠ Running the `uninstall` will not delete `dfs.data.dir`, `dfs.name.dir`, `dfs.mapred.dir` and `dfs.checkpoint.dir` directories. This is intentional behavior and preserves user data.

```
[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
```

**Sample Usage**

```
icm_client uninstall -l CLUSTERNAME
```

⚠ If you had HAWQ installed as part of the cluster, you will have to manually reset the `limits.conf` and `sysctl.conf` files on the HAWQ nodes before you can reuse those nodes again.

# Managing HAWQ

Starting and stopping HAWQ can only be initiated directly on the HAWQ Master. More information about HAWQ can be found in the *Pivotal HAWQ 1.x Installation Guide* and the *Pivotal ADS 1.x Administrator Guide*.

## Initializing HAWQ

You must initialize HAWQ only once after the cluster has started and specifically after the HDFS is up and running:

```
[gpadmin]# source /usr/local/hawq/greenplum_path.sh
[gpadmin]# /etc/init.d/hawq init
```

Running the `init` command, completes the following:

- Initializes the HAWQ master and the segment hosts.
- Starts the HAWQ master, segments, and the underlying postgres database.
- Exchanges SSH keys between the Master and Segment nodes.

- This operation takes a few minutes to complete.
- If you need to initialize the HAWQ standby master refer to the *Pivotal HAWQ Installation Guide* for instructions

If you have a HAWQ Standby master in your cluster configuration, initialize that by running the following:

```
# gpinitstandby -s <HAWQ STANDBY MASTER FQDN>
```

## Starting HAWQ

Run the `start` command to start up the HAWQ master and all the segments hosts including the postgres database.

Note that this is implicitly done as part of the HAWQ Initialization.

```
[gpadmin]# /etc/init.d/hawq start
```

## Stopping HAWQ

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. Logon to the HAWQ Master as user `gpadmin`.
2. In the `$MASTER_DATA_DIRECTORY/pg_hba.conf` (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.
   Find the entry:
   `host all gpadmin <master_host_ip>/32 trust`
   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)
- Manually editing the `hawq-segments` section of the `clusterConfig.xml` file, then running the `reconfigure` command (see Reconfiguring a Cluster)

Once you have added the slaves, you can then expand HAWQ using the `gpexpand` command; refer to the *HAWQ Administration Guide - Expanding the HAWQ System* for details.

# Managing 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 the following options when managing cluster and individual roles:

- - Managing locally
  - Managing from the Admin Node

## Managing Locally

You can manage the service role on the target host locally. For example, to restart 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 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 | |

| Role Name | Service Command |
|---|---|
| | `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}` |

# Pivotal HD 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.checkpointdir* | 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 |
| | *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 |

| Service | Directory | Owner | Permissions |
|---|---|---|---|
| | *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

If you are running a firewall, ensure that the following ports are open

| Service | Port |
|---|---|
| ssh | 22 |
| NameNode | 8020 (Apache 9000) |
| NameNode Web UI | 50070, 50470 (https) |
| Secondary NameNode | 50090 |
| DataNode Communication | 50010 |
| DataNode IPC | 50020 |
| DataNode HTTP Address | 50075 |
| ResourceManager Web UI | 8042,8088 |
| ResourceManager | 8030,8031,8032,8033 |
| MapReduce Shuffle Port | 7070 |
| Job History Server | 10020 |
| Job History Web UI | 19888 |

| Service | Port |
|---|---|
| JobTracker | (Apache 9001) |
| JobTracker Web UI | (Apache 50030) |
| TaskTracker | (Apache 50060) |
| Puppet | 443,8140,61613 |
| Jetty | 8080 |
| HBase Master | 60000 |
| HBase Master UI | 60010 |
| HBase RegionServer | 60020 |
| HBase RegionServer Web UI | 60030 |
| ZooKeeper Client | 2181 |
| ZooKeeper Leader | 3888 |
| ZooKeeper Peers | 2888 |
| HAWQ Master | 8432 |
| HAWQ Port Base | 40000 |
| Quorum Journal node port | 8485 |

# Chapter 8 PHD FAQ (Frequently Asked Questions)

**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?**

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
```

3. 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
```

4. Start Pivotal Command Center again:

```
service commander start
```

# Chapter 9 PHD Troubleshooting

This section provides common errors you may receive and how to troubleshoot or workaround those errors.

**Topics:**

- Resolving postgres port Conflict Issue
- Resolving HTTP Port Conflict
- Errors like Ambit: Push Failed
- Preparehosts Errors Out While Creating gpadmin User
- HAWQ Initialization Failing
- Installing HAWQ on Dirty Cluster Nodes Previously Configured with HAWQ
- Errors Related to VM Memory

# 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 PFX configured as a service, you must manually reinstall those services. See Upgrading PHD Using the CLI 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 to RPM 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 Failure without Generating a Log 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/gphdmgr
   ```

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 *Pivotal Command Center Installation and 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 (refer to `prepare-hawq-hosts`) 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 the *HAWQ Administator Guide* 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 intializing 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 0 to 2:

  ```
  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).