Pivotal™

PRODUCT DOCUMENTATION

Pivotal HD Enterprise

Version 1.1

Installation and Administrator Guide

Rev: A03

Table of Contents

1 Overview of Pivotal HD Enterprise

Pivotal HD Enterprise is an enterprise-capable, commercially supported distribution of Apache Hadoop 2.0 packages targeted to traditional Hadoop deployments.

This overview describes each component and how it fits into the overall architecture of Pivotal HD Enterprise.

1.1 Components

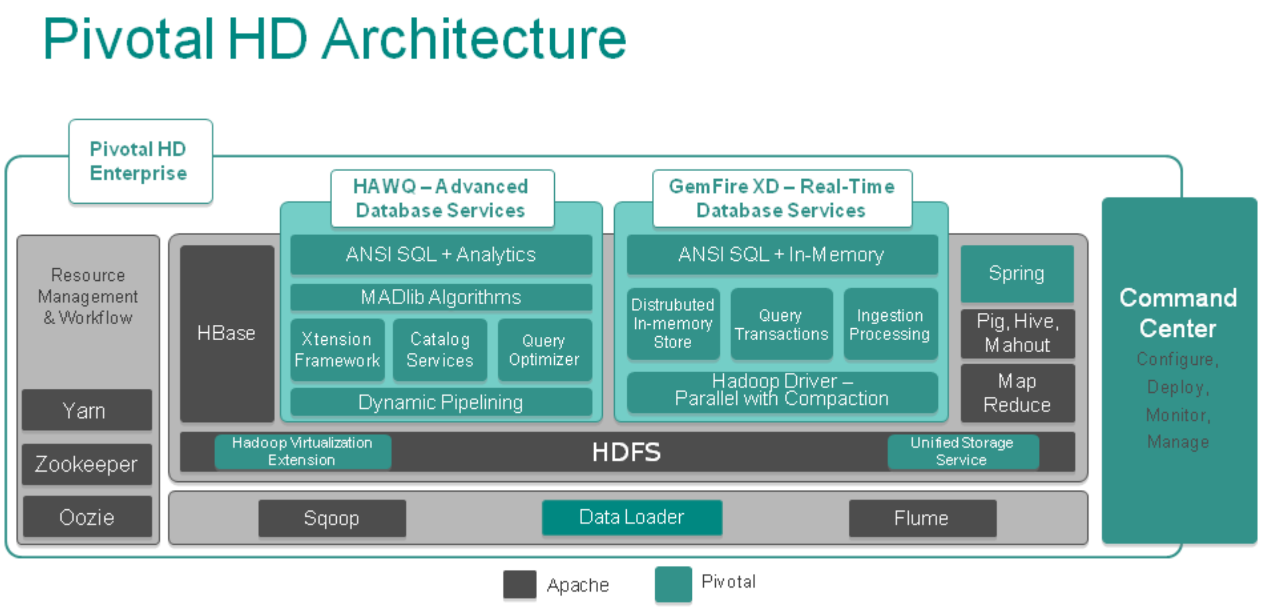
Pivotal HD Enterprise is a commercially-supported distribution of the Apache Hadoop stack including the following:

- **Core Apache Stack**:
  - Hadoop (MR2)
    - HDFS
    - YARN
  - Hadoop (MR1)
    - MapReduce
  - Zookeeper
  - HBase
  - Hive
  - Pig
  - Mahout
  - Flume
  - Sqoop
  - HVE
  - 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.
- **PHD Tools**
  - **USS** - Unified Storage System, a framework that provides HDFS Unified Storage System, a framework that provides HDFSprotocol layer on top of external file systems.
  - **DataLoader** - High-speed data ingest tool for your Pivotal HD cluster.

- **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.
- **PRTS - Pivotal Real Time Services** - Pivotal HD 1.1 includes support for GemFire XD (GFXD) Beta, an offering of PRTS. For further information about GemFire XD installation and configuration; refer to the section Configuring GemFire XD Beta .

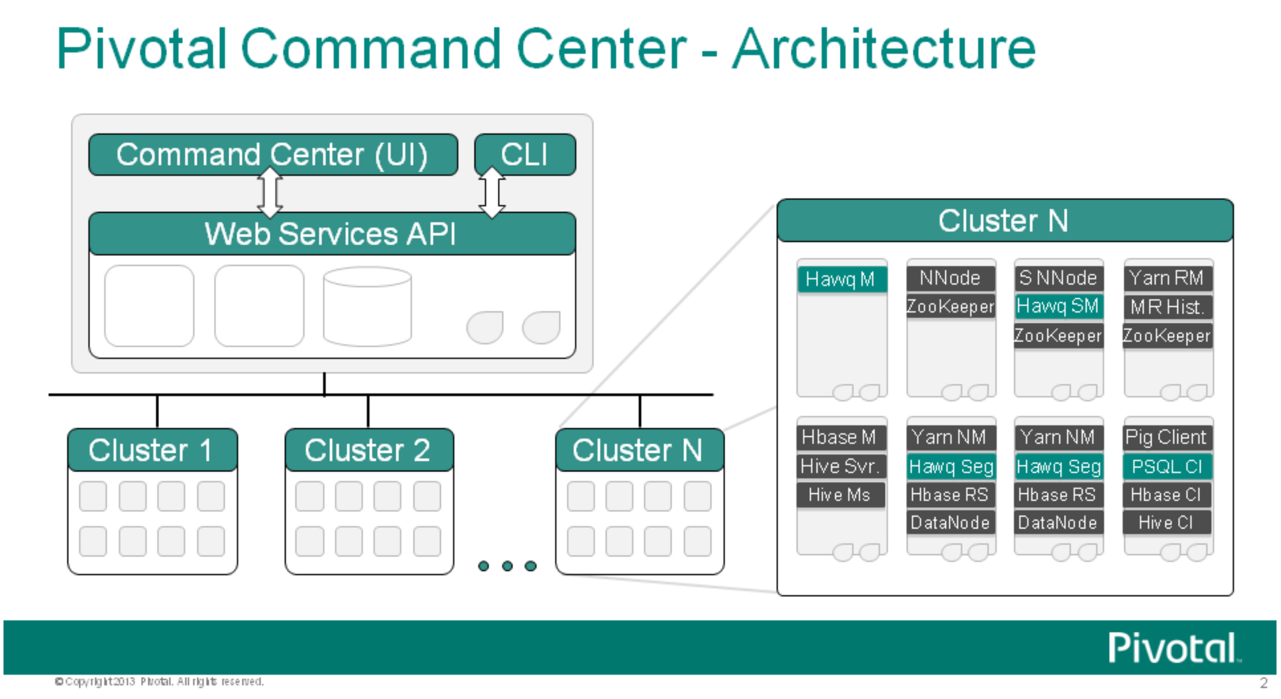


Pivotal Command Center (PCC) includes a CLI (command line interface) and a GUI. You can deploy and configure most of the Hadoop services as well as HAWQ, PXF and USS, using the either the CLI or the GUI. You can start and stop the clusters using either the CLI or the GUI.

This documentation covers operations performed via the CLI. For Pivotal Command Center GUI operations; including configuring and deploying clusters, see the *Pivotal Command Center 2.1 User Guide*.

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

See Deployment Options.



1.2 About Supported Pivotal HD Services

The following services can be deployed and configured via Pivotal Command Center CLI.

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

The following services can be deployed and configured manually (see the *Pivotal HD Enterprise 1.0 Stack and Tool Reference Guide* for details)

- Flume
- Sqoop
- Oozie (YARN-based clusters only)

Pivotal HD 1.1 supports YARN (MR2) resource manager by default. If you don't want to deploy a YARN based cluster, we provide MR1 as optional manually-installable software, instructions for which are provided in the *Pivotal HD Enterprise 1.0 Stack and Tool Reference Guide*.

Note that since MR1 needs to be installed manually, you won't be able to use Pivotal Command Center for monitoring and management of the cluster.

HDFS

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

The following table shows HDFS service roles:

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

Notes

- HBase requires that you have installed HDFS, YARN, and Zookeeper.
- Pivotal HD installs ZooKeeper if you have not installed it.
- Pivotal HD only supports Distributed Mode (Not Standalone Mode)
- Pivotal HD does not install the HBase Thrift Server
- 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

Role Name	Description
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 1.0 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.

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.

USS

USS (Unified Storage System) is a abstraction layer for HDFS-like storage systems that allows users access to a multitude of storage systems under a single namespace.

Role Name	Description
USS Agent	This role is present on all datanodes/nodemanagers in the cluster and installs the USS library on them
USS Namenode	The USS Namenode is a Hadoop RPC server capable of resolving USS URIs containing mount-points to their actual URIs on delegate FileSystems
USS Catalog	The catalog stores the metadata for USS mountpoints. Postgres database is used as the datastore.
USS Client	This is a launcher or gateway node for USS administrators to manage the USS catalog. It is installed on client nodes configured in your cluster.

Note: USS requires HDFS

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.

Sqoop

Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. For more details, please refer to the Apache Sqoop page: http://sqoop.apache.org/

Oozie

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

2 Installing Pivotal HD Enterprise Using the CLI

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

2.1 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, Pig, and USS with default settings. • Reconfigures the supported cluster services. • Multi-cluster support. • Monitors clusters with Pivotal Command Center (PCC).
Starting and stopping	• Starts and stops the cluster or individual services. • Ensures that all dependent services start and stop in the correct order.
Logging	• Provides installation data logs.
Uninstallation	• Can uninstall individual services and Pivotal HD Enterprise.

2.2 Planning your Pivotal HD Cluster Deployment

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

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

2.2.1 Deployment Options

Pivotal HD 1.1 supports YARN (MR2) by default. If you don't want to deploy a YARN-based cluster, we provide Hadoop MR1 as optional manually-installable software.

MR1-based Clusters

- You can only use the components provided in the MR1 package; you cannot combine a stack component from an MR1 package with one from an MR2 package.
- You cannot install MR1 components using Pivotal Command Center's CLI installation utility.You must install MR1 components manually (both rpm and bin installations are supported), see the Pivotal HD Enterprise Stack and Tool Reference Guide for details.

The following table illustrates the deployment options and limitations:

Component	CLI install	Manual install (rpm)	Manual install (bin)
Command Center (installs the CLI)		✓	
Hadoop MR2: HDFS, YARN	✓	✓	✓
Pig	✓	✓	✓
Hive	✓	✓	✓
HBase	✓	✓	✓
Mahout	✓	✓	✓
Zookeeper	✓	✓	✓
Flume		✓	✓
Hcatalog		✓	✓
Sqoop		✓	✓
HVE		✓	✓
Oozie			✓

Component		CLI install	Manual install (rpm)	Manual install (bin)
Advanced Database Services:	HAWQ	✓	✓	✓
	PXF	✓	✓	✓
PHD Tools	USS	✓	✓	✓
	DataLoader		✓	✓

2.2.2 Installation Instructions:

You can find installation instructions for the above components in these documents:

- Pivotal Command Center: *Pivotal HD Enterprise Installation and Administrator Guide* (this document)
- Hadoop MR2 (via CLI): *Pivotal HD Enterprise Installation and Administrator Guide* (this document)
- Hadoop MR2 and MR1 manual installs: *Pivotal HD Enterprise 1.1 Stack and Tool Reference Guide*.
- USS: *Pivotal HD Enterprise 1.1 Stack and Tool Reference Guide*.
- Pivotal DataLoader: *Pivotal HD 2.0 Installation and Administrator Guide*.

2.2.3 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 and so designing for a specific type of workload may 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 & Memory, as workload evolves over the time and demands for it.
Hardware and capacity requirements for cluster nodes may 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. So it is important to provide more reliable hardware for master nodes (such as NameNode, YARN Resource manager, HAWQ master) for higher cluster availability.

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

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

Cluster Slaves

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

- 2 CPUs (4 to 8 cores)--- You can also have single CPU with more (6 to 8) cores and ability to add additional CPU, 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 task per node when using PHD with HAWQ and assigning more cores to HAWQ segment servers based on mix work load of HAWQ vs. MapReduce.
- 24 to 64GB RAM per node — Typically 1 GB for each Hadoop daemons 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 requirement.
- 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 may put more memory requirement on the HDFS Namenode. The reason for this is the total HDFS storage capacity grows with more 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 more reliable hardware for cluster master nodes.

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

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

Pivotal HD Admin node

Ensure that the Admin node is separate from cluster nodes especially if cluster size 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

2.2.4 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, here are some guidelines for better performance, availability, and use:

- Hadoop services Master roles: For example, HDFS NameNode, YARN ResourceManager and History Server, HBase Master, HAWQ Master, USS Namenode. 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 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 a bunch of systems with defined roles (admin node, namenode, HAWQ master, etc) all ready for install/deploy of the PHD software distribution.

2.3 Command Line Installation Overview

The table below provides a brief overview of the installation steps. Each step is covered in more detail in the following sections of this document.

Task	Subtasks
Prerequisites See Pivotal HD Prerequisites for more details	**Check JDK version** (as root) `# java -version` Ensure you're running Oracle Java JDK Version 1.7. If not, download the appropriate version from Oracle. Note: JDK 1.6 is optional but has not been fully tested.
	Check Yum accessibility (as root) Verify that all hosts have yum access to an EPEL yum repository. `# sudo yum list < LIST OF PACKAGES >` See Pivotal HD Prerequisites for the list of packages.

Task	Subtasks

	Verify iptables is turned off
	(as root)
	`# chkconfig iptables off`
	`# service iptables stop`
	`# service iptables status`
	`iptables: Firewall is not running.`
	
	Note: All machines in the cluster must also allow ICMP between boxes, and the admin server must respond to ping. This is used during the `icm_client scan hosts`' command to test that the nodes can reach the admin server.

	Disable SELinux (as root) `# echo 0 >/selinux/enforce`

Prepare the Admin Node
See Preparing the Admin Node for more details
	Install Pivotal Command Center
	(as root)
	1. Copy tar file to your specified directory on the admin node, for example:
	`# scp ./PCC-2.0.x. version.build.os .x86_64.tar.gz host:/root/phd/`
	
	2. Login as root and untar to that directory:
	`# cd /root/phd`
	`# tar --no-same-owner -zxvf PCC-2.0.x. version.build.os .x86_64.tar.gz`
	3. Run the installation script from the directory where it is installed:
	`# ./install`
	4. As the rest of the installation is done as the `gpadmin` user, change to `gpadmin` user:
	`# su - gpadmin`
	
	5. Enable Secure Connections

	Import the Pivotal HD, HAWQ and PHDTools packages to the Admin node (as gpadmin) 1. Copy the Pivotal HD, ADS (HAWQ), and PHDTools (optional for USS) tarballs from the initial download location to the gpadmin home directory 2. Change the owner of the packages to gpadmin and untar the tarballs **Note**: If you want to use GemFire XD Beta, you also need to import and enable the PRTS package. Complete instructions are in the Configuring GemFire XD Beta section.
	Enable Pivotal HD Service (as gpadmin) `# icm_client import -s < PATH TO EXTRACTED PHD TAR BALL >`

Task	Subtasks
	Enable HAWQ and PXF services (as gpadmin) `# icm_client import -s < PATH TO EXTRACTED ADS TAR BALL >`
	Enable USS services Optional (as gpadmin) `# icm_client import -s < PATH TO EXTRACTED PHDTOOLS TAR BALL >`
Edit the Cluster Configuration File See Cluster Configuration Files for more details	**Fetch the default Cluster Configuration template** (as gpadmin) `# icm_client fetch-template -o ~/ClusterConfigDir` Note: `ClusterConfigDir` will be automatically created.
	Edit the default Cluster Configuration template (`clusterConfig.xml`) (as gpadmin) At a minimum, you must replace all instances of `host.yourdomain.com` with valid hostnames for your deployment. **Notes**: 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 Beta section. 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.
	Configure other Pivotal HD and ADS Components (as gpadmin) Optional: Configure HAWQ and other stack components in their corresponding configuration files (for example: `hawq/gpinitsystem_config` file), as needed
	Configure USS (as gpadmin) Optional: Configure USS, (configuration files located in `/uss`), by defining the **uss-catalog** role in the `clusterConfig.xml` file for every cluster.

Task	Subtasks
Deploy the Cluster See Deploying the Cluster for more details	**Deploy/Install a cluster** (as gpadmin) `# icm_client deploy -c ~/ClusterConfigDir` NOTE: This command creates the gpadmin user on the cluster nodes. Do NOT create this user manually. If gpadmin already exists on the cluster nodes, delete that user before running this command.

	Post installation for HAWQ
	(as gpadmin)
	Exchange keys between HAWQ master and segment hosts:
	Create a hostfile (`HAWQ_Segment_Hosts.txt`) that contains the hostnames of all your HAWQ segments.
	
	`# ssh < HAWQ_MASTER`
	`# source /usr/local/hawq/greenplum_path.sh`
	`# /usr/local/hawq/bin/gpssh-exkeys -f ./HAWQ_Segment_Hosts.txt`

Verify the Cluster See Verifying a Cluster for more details	**Start the Cluster** (as gpadmin) `# icm_client start -l < CLUSTERNAME >`

	Verify HDFS
	(as gpadmin)
	
	`# ssh < NAMENODE >`
	`# hdfs dfs -ls /`

	Initialize HAWQ (as gpadmin) ssh to the HAWQ master, the run the following: `# source /usr/local/hawq/greenplum_path.sh # /etc/init.d/hawq init`

2.4 Pivotal HD Enterprise Prerequisites

1. 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

2. **DNS lookup**. Verify that the admin host is be able to reach every cluster node using its hostname and IP address. Verify 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
```

3. **iptables**. Verify that iptables is turned off:

As root:

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

Note: All machines in the cluster must also allow ICMP between boxes, and the admin server must respond to ping. This is used during the `icm_client scan hosts`' command to test that the nodes can reach the admin server.

4. **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`
Reboot the system.

5. **JAVA JDK**. Ensure that you are running Oracle JAVA JDK version 1.7.

Note: Oracle JDK 1.6 is optional but not fully tested. Customers may use JDK 1.6 but will not receive official support.

As root:

```
# java -version
```

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

6. **YUM**. Verify that all hosts have yum access to an EPEL yum repository. See Package Accessibility, below.

2.4.1 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 would 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), 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

- 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:

```
$ sudo yum list <LIST OF PACKAGES>
```

If any of them are not available or not already installed, then you may have not added the repository correctly 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

Back to Installation Overview

2.5 Preparing the Admin Node

1. Make sure the following tarballs are available on the Admin node:

- *Pivotal Command Center (PCC)
- *Pivotal HD tarball (Apached Hadoop related services)
- *Pivotal ADS tarball (HAWQ, PXF services)
- Oracle JDK 1.7 Package - (you can download this from the Oracle site at http://www.oracle.com/technetwork/java/javase/downloads/index.html)

Note: Oracle JDK 1.6 is optional but not fully tested. Customers may use JDK 1.6 but will not receive official support.

*You can download these packages from the EMC Download Center at https//emc.subscribenet.com.

2. Make sure that all the tarballs are extracted and readable by the `gpadmin` user.

2.5.1 Install Pivotal Command Center

1. Copy the PCC tar file to your specified directory on the admin node, for example:

```
# scp ./PCC-2.0.x.version.build.os.x86_64.tar.gz host:/root/phd/
```

2. Login as root and untar to that directory:

```
# cd /root/phd
     # tar --no-same-owner -zxvf PCC-2.0.x.version.build.os.x86_64.tar.gz
```

3. Run the installation script from the directory where it is installed:

```
# cd PCC-2.0.x.version
      # ./install
```

   This installs the required packages and configures Pivotal Command Center and starts PCC services.
4. Enable Secure Connections:

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 PCC configuration file `/usr/local/greenplum-cc/config/commander` as follows:

Change the path referenced in the variable `PCC_SSL_KEY_FILE` to point to your own key file.

Change the path referenced in the variable `PCC_SSL_CERT_FILE` to point to your own certificate file.

⚠ See SSL Certificates for details

d. Restart PCC with the following command:

```
service commander restart
```

5. Verify that your PCC instance is running by executing the following command:

```
$ service commander status
```

The PCC installation also includes a CLI (Command Line Interface tool, `icm_client`). You can now deploy and manage the cluster using the CLI.

From now on you can switch to the gpadmin user. You should no longer need to be root for anything else.

```
su - gpadmin
```

2.5.2 Install Pivotal HD, PHDTools, and HAWQ

Once you have Pivotal Command Center installed (the Command Center installation includes a CLI tool, *icm_client*, you now use to deploy and configure PHD services), you can use the *import* utility to sync the RPMs 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 RPMs during deployment.

Note: If you want to use GemFire XD Beta, you also need to import and enable the PRTS package. Complete instructions are in the Configuring GemFire XD Beta section.

Note: Run *import* each time you wish to sync/import a new version of the package.

1. Copy the Pivotal HD, ADS, and PHDTools tarball from the initial download location to the gpadmin home directory

2. Change the owner of the packages to gpadmin and untar both the tarballs. For example:

```
If the file is a tar.gz or tgz, use
tar zxf PHD-1.0.x-x.tgz

If the file is a tar, use
tar xf PHD-1.0.x-x.tar

Similarly for the Pivotal ADS tar.gz or tgz file, use
tar zxf PADS-1.0.x-x.tgz

If the file is a tar, use
tar xf PADS-1.0.x-x.tar

Similarly for the PHDTools tar.gz or tgz file, use
tar zxf PHDTools-1.0.x-x.tgz

If the file is a tar, use
tar xf PHDTools-1.0.x-x.tar
```

Enabling Pivotal HD Service

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

```
# icm_client import -s <PATH TO EXTRACTED PHD TAR BALL>
```

Example:

```
# icm_client import -s PHD-1.0.x-x/
```

Enabling HAWQ and PXF Services

Note: This is required only if you wish to deploy HAWQ.

1. As gpadmin, extract the following tar ball for HAWQ and PXF:

```
# icm_client import -s <PATH TO EXTRACTED ADS TAR BALL>
```

Example:

```
# icm_client import -s PADS-1.0.x-x/
```

For more information, see the log file located at: `/var/log/gphd/gphdmgr/gphdmgr-import.log`

Enabling USS Service

Note: This is required only if you wish to deploy and enable USS.

1. As gpadmin, extract the following tar ball for USS:

```
# icm_client import -s <PATH TO EXTRACTED PHDTools TAR BALL>
```

Example:

```
# icm_client import -s PHDTools-1.0.x-x/
```

Syntax:

```
icm_client import --help
Usage: icm_client [options]

Options:
  -h, --help            Show this help message and exit
  -p PATH, --path=PATH  (deprecated: use -s instead) directory location of
                        PHD/PADS/PHDTools/PRTS stack
  -v VERSION, --version=VERSION
                        The version of PHD stack - 1.0 or 2.0 (default)
  -s STACK, --stack=STACK
                        Directory location of the PHD/PADS/PHDTools/PRTS stack
  -r RPM, --rpm=RPM     Location of the rpm to be included in the PHDMgr local
                        yum repository
  -f FILE, --file=FILE  Location of the file like JDK-version.bin which will
                        be used by PHDMgr during cluster deployment
```

Note: The version value [`-v`] is an optional field and defaults to "2.0". If at all you specify it, do not use the actual package version, use "2.0". The 2.0 corresponds to the Apache YARN (2.x) version

Back to Installation Overview

2.6 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 `host.yourdomain.com` 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`*.*

Specifically, for HAWQ you may have to edit the HAWQ configuration file, see Configuring HAWQ; and iff you want to enable HA, you need to make some HA-specific changes to several configuration files. Complete instructions are available in the High Availability section.

2.6.1 Fetching Default Cluster Configuration Templates

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

1. As gpadmin, run the `fetch-template` command.

Example:

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

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

Syntax:

```
icm_client fetch-template --help
Usage: /usr/bin/icm_client fetch-template options

Options:
-h, --help          Show this help message and exit
-o OUTDIR, --outdir=OUTDIR
                    Directory path to store the cluster configuration template files
```

2.6.2 Editing Cluster Configuration

1. Locate and update the `clusterConfig.xml` file based on your cluster requirements. At a minimum, you must update the default names of the nodes in this file to match the names of the nodes in your own cluster.
   Notes:
   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 Beta section.
   If you want to enable HA, you need to make some HA-specific changes to the `clusterConfig.xml` file and 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
   ```

About the Cluster Configuration File

This section provides more information about what is contained in the Cluster Configuration file and what you can edit based on your cluster.

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
hbase
    hadoop-metrics.properties
    hbase-env.sh
    hbase-policy.xml
    hbase-site.xml
    log4j.properties
hawq
    gpinitsystem_config
pig
    log4j.properties
    pig.properties
hive
    hive-env.sh
    hive-exec-log4j.properties
    hive-log4j.properties
    hive-site.xml
uss
    uss.properties
    uss-client-site.xml
    uss-env.sh
    uss-nn-site.xml
```

Note: There may not be a folder that corresponds to every service, for example Pig and Mahout do not have their own directories, they can be configured directly using the client tag in the `clusterConfig.xml` file.

The `clusterConfig.xml` file contains the following sections:

Head Section

This is the metadata section and must contain the following mandatory information:

- `clusterName`: Configure the name of the cluster
- `gphdStackVer`: Pivotal HD Version. This defaults to 2.0

- `services`: Configure the services to be deploy. 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, Mahout jobs.

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. Please 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.

> **Important**
>
> **If you are planning to configure Hive with HAWQ/PXF, please ensure that the "hive-server" is co-located with the Hadoop "namenode". This requirement is due to a known bug and will be fixed in future releases**
>
> **We recommend you use FQDN instead of short hostnames in the clusterConfig.xml file**

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 respectively. These properties can be found in the 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 mount points mentioned in this section are automatically created by Pivotal HD during cluster deployment.
> - We recommend you have multiple disk mount points for datanodes, but it is not a requirement.

Other Configuration Files

Please ensure that the directories specified for `dfs.datanode.name.dir` and `dfs.datanode.data.dir` in the `hdfs/hdfs-site.xml` are empty.

Configuring Your Hadoop Service

Each service has a corresponding directory that containing standard configuration files. You can override properties to suit your cluster requirements, or consult with Pivotal HD support to decide on a configuration to suite your specific cluster needs.

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

```
Example In hdfs/core-site.xml: fs.defaultFS which is set to

hdfs://$<NAMENODE>:$<dfs.port>
```

2.7 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

If you are planning to configure Hive with HAWQ/PXF, please ensure that the "hive-server" is co-located with the Hadoop "namenode". This requirement is due to a known bug and will be fixed in future releases.

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

Remove the entry `vm.overcommit_memory = 2` from `/usr/lib/gphd/gphdmgr/hawq_sys_config/sysctl.conf` prior to running the `prepare hawq` utility.

In the `clusterConfig.xml`, update `<hawq.segment.directory>` to include only one segment directory entry (instead of the default 4 segments).

Back to Installation Overview

2.8 Configuring USS

The USS configuration files are located in the `uss/` directory.

The yarn configuration file needs to be updated, as follows:

In `/etc/gphd/hadoop/conf/yarn-site.xml` update the `yarn.application.classpath` property to include the following classpath entries:

```
$USS_HOME
$USS_CONF
```

Pivotal HD allows users to deploy USS in a couple of different scenarios.

2.8.1 Deploy USS on a single cluster

This use case is supported by deploying the `uss-catalog` for the current cluster

This is done by including the **uss-catalog** rule in the clusterConfig.xml file for every cluster.

2.8.2 Deploy USS across a set of clusters deployed by Pivotal HD

This use case is supported by not deploying the `uss-catalog` for the current cluster, but using `uss-catalog` from a cluster deployed previously via the CLI.

To do this perform the following:

1. Deploy a dedicated cluster containing the `uss-catalog` role, for example, Cluster-1.

2. Deploy subsequent clusters without the `uss-catalog` role, for example, Cluster-2 and Cluster-3. While deploying Cluster-2 and Cluster-3

a. Remove the `uss-catalog` from `clusterConfig.xml`.

b. Update `uss.db.url` `uss/uss.properties` with the hostname and port of the `uss-catalog` in Cluster-1 cluster configuration.

2.9 Verifying the Cluster Nodes for Pivotal HD

1. As gpadmin, run the `scanhosts` command to verify certain prerequisites are met.

Example:

```
# icm_client scanhosts -f ./HostFile.txt
```

The `scanhosts` command verifies that prerequisites for the cluster node and provides a detailed report of any missing prerequisites. Running this command ensures that clusters are deployed smoothly.

Syntax:

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

Options:
  -h, --help            Show this help message and exit
  -v, --verbose         Increase output verbosity
  -f HOSTFILE, --hostfile=HOSTFILE
                        File containing new-line separated list of hosts to be
                        scanned
  -j JAVAHOME, --java_home=JAVAHOME
                        java_home path configured
```

You can troubleshoot using the following files:
On the Admin Node:

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

On the Cluster Nodes:

- `/var/log/gphd/gphdmgr/ScanHost.XXX.log`

Note: We recommend running the scanhosts before every deployment or reconfiguration of the cluster.

2.10 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 and the installation procedure.

Deploy the cluster as gpadmin.

The deploy command internally does three steps:

1. Prepares the cluster nodes with the pre-requisites (runs `preparehosts` command)
2. Verifies the prerequisites (runs `scanhosts` command)
3. Deploys the cluster

Note: Ensure that the JDK file downloaded from Oracle has execute permission.

The `preparehosts` command run internally as part of deploy performs the following on the hosts listed in the `clusterConfig.xml` file:

- Creates the `gpadmin` user.
- As `gpadmin`, sets up password-less SSH access from the Admin node.
- Installs the provided Oracle Java JDK.
- Disables SELinux across the cluster.
- Synchronizes the system clocks.
- Installs Puppet version 2.7.20 (the one shipped with the PCC tarball, not the one from puppetlabs repo)
- Installs sshpass.

If Oracle JDK is not already installed on the cluster nodes, you can install it via the deploy command. Here are the steps to deploy JDK.

1. Accept the license agreement and download Oracle JDK from the Oracle website
2. Import the JDK file into the PHDMgr files repo. PHDMgr looks for the files in its repository during deployment:

   Note: Oracle JDK 1.7 is recommended. Oracle JDK 1.6 is optional but not fully tested. Customers may use JDK 1.6 but will not receive official support.

   If you have downloaded JDK 1.7, it will be a `.rpm` file. Use the following command to import the `.rpm` file:
   `icm_client import -r <JDK rpm>`
   If you have downloaded JDK 1.6, it will be a `.bin` file. Use the following command to import the `.bin` file`icm_client import -f <JDK bin>`
3. Once the file is imported, you can just use the name of the file in the `-j` option of `preparehosts` during deployment.

! Pivotal recommends that you run only one deploy command at a time, because running simultaneous deployments might result in failure.

Example:

```
# icm_client deploy -c clusterConfigDir/ -i -d -j jdk-6u43-linux-x64-rpm.bin
```

! Use the `-t` option only if you have NTP setup on the nodes.

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     Donot verify cluster nodes as part of deploying the
                        cluster
  -p, --nopreparehosts  Donot prepare hosts as part of deploying the cluster
  -j JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM installer binary (Ex:
                        jdk-6u41-linux-x64-rpm.bin). Ignored if -p is
                        specified
  -t, --ntp             Synchronize system clocks using NTP (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
```

The deploy command has the option to skip running `preparehosts` or `scanhosts` as part of the deployment. If you have manually run `preparehosts` and `scanhosts` commands to confirm that all pre-requisites are met then you can skip running them during deployment using the `-s` and `-p` options.

You can also specify options to the preparehosts using `-j`, `-t`, `-d`, or `-i` options.

2.10.1 Post Installation for HAWQ

You need to exchange SSH keys between HAWQ Master and Segment Nodes to complete HAWQ installation.

1. Create a hostfile (`HAWQ_Segment_Hosts.txt`) that contains the hostnames of all your HAWQ segments.

2. As gpadmin, execute the following commands **from the HAWQ Master**.

```
# ssh <HAWQ_MASTER>
# source /usr/local/hawq/greenplum_path.sh
# /usr/local/hawq/bin/gpssh-exkeys -f ./HAWQ_Segment_Hosts.txt
```

Your Pivotal HD installation is now complete.

You can now start a cluster and start HAWQ. Both these steps are described in "Verifying a Cluster Installation", next.

Back to Installation Overview

2.11 Verifying a Cluster Installation

We recommend that you verify your cluster installation.

To verify your cluster installation:

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.

2.Verify HDFS is running (you will not be able to initialize HAWQ if HDFS is not running)

```
# hdfs dfs -ls/
```

Sample Output:

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

3. 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:

```
# source /usr/local/hawq/greenplum_path.sh
# /usr/local/hawq/bin/gpssh-exkeys -f ./HAWQ_Hosts.txt
# /etc/init.d/hawq init
```

See Managing HAWQ sections for more detailed instructions.

2.11.1 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'
```

2.12 Preparing the Cluster Nodes (Deprecated)

This is an optional command that can be used to prepare the hosts before a cluster deployment. This command is internally run as part of deploy so it not necessary that you run this before a cluster deployment. This command has been retained for backward compatibility.

1. Create a hostfile (`HostFile.txt`) that contains the hostnames of all your cluster nodes except the Admin node; separated by newlines. You will have to input this file in many Pivotal HD commands.

For example, the hostfile should look like the following:

```
[gpadmin] cat HostFile.txt
host1.pivotal.com
host2.pivotal.com
host3.pivotal.com
```

Note: The following script shows how to create a hostfile as input for a large number of hosts:

```
[gpadmin] for i in `seq \-w 1 3`; do sudo sh \-c "echo \"host$i.pivotal.com\" >> HostFile.txt";
done
```

Important: The hostfile should contain all nodes within your cluster EXCEPT the Admin node.

2.12.1 Preparing the Cluster Nodes for Pivotal HD

1. As gpadmin, run the `preparehosts` command to perform some administrative tasks to prepare the cluster for Pivotal HD.

Note: One of the tasks this command performs is to create the gpadmin user on the cluster nodes. Do NOT create this user manually. If gpadmin user already exists on the cluster nodes, delete that user by running:

```
pkill -KILL -u gpadmin
userdel -r gpadmin
```

Run `preparehosts`:

Example:

```
# icm_client  preparehosts --hostfile=./HostFile.txt --java=<PATH TO THE DOWNLOADED SUN JAVA
JDK> --ntp --selinuxoff --iptablesoff
```

⚠ **Ensure that the JDK file downloaded from Oracle has execute permission.**

The `preparehosts` command performs the following on the hosts listed in the hostfile:

- Creates the `gpadmin` user.
- As gpadmin, sets up password-less SSH access from the Admin node.
- Installs the provided Oracle Java JDK Version 1.6.
- Disables SELinux across the cluster.
- Synchronizes the system clocks.
- Installs Puppet version 2.7.20 (the one shipped with the PCC tarball, not the one from puppetlabs repo)
- Installs sshpass.

Do not use the Admin node as part of your cluster. The preparehosts command will automatically remove the admin host if included. It will not prepare the admin node as it might corrupt the admin nodes' certification process that is part of the puppet orchestration during deploy.

Syntax:

```
icm_client preparehosts --help
Usage: icm_client [options]

Options:
  -h, --help            Show this help message and exit
  -f HOSTFILE, --hostfile=HOSTFILE
                        File containing a list of all cluster hosts (newline
                        separated)
  -j JAVA, --java=JAVA  Location of Sun Java JDK RPM installer binary (Ex:
                        jdk-6u41-linux-x64-rpm.bin)
  -t, --ntp             synchronize system clocks using NTP (requires external
                        network access)
  -d, --selinuxoff      disable SELinux
  -i, --iptablesoff     disable iptables
```

2.12.2 Preparing the Cluster Nodes for HAWQ

This command has to be run only if you plan to install HAWQ. The `prepare-hawq-hosts` command sets kernel parameters that optimize HAWQ performance. In particular, this utility modifies the `/etc/sysctl.conf and /etc/security/limits.conf` file. The recommended configurations are available in the `/usr/lib/gphd/gphdmgr/hawq_sys_config/` file on the Admin node.

1. Create a hostfile (`HAWQ_Hosts.txt`) that contains the hostnames of all your HAWQ nodes (HAWQ master, standby master, and segment nodes); separated by newlines.

2. As gpadmin, run the `prepare-hawq-hosts` command to optimize HAWQ performance.

Example:

```
# icm_client prepare-hawq-hosts -f ./HAWQ_Hosts.txt -g /usr/lib/gphd/gphdmgr/hawq_sys_config/
```

Notes:

- The hostfile must contain all the HAWQ nodes (HAWQ master, standby master and segment nodes).

- The command prepare-hawq-hosts edits the limits.conf and sysctl.conf files. Pivotal recommends you review these configurations before you run the command.
- If you are planning to deploy a HAWQ cluster on a VM with memory lower than the optimized/recommended requirements:

Remove the entry vm.overcommit_memory = 2 from /usr/lib/gphd/gphdmgr/hawq_sys_config/sysctl.conf prior to running the prepare hawq utility.

In the clusterConfig.xml, update <hawq.segment.directory> to include only one segment directory entry (instead of the default 4 segments).

Syntax:

```
icm_client prepare-hawq-hosts  -h
Usage: icm_client [options]

Options:
  -h, --help            Show this help message and exit
  -f HOSTFILE, --hostfile=HOSTFILE
                        File containing a list of all cluster hosts where HAWQ
                        will be installed (newline separated)
  -g GPCC, --gpcc=GPCC  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
```

Back to Installation Overview

2.13 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.

Directory Location	Description
`/usr/bin`	The folder for the component's command scripts; only sym-links or wrapper scripts are created here.

2.14 Running Sample Programs

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

2.14.1 Testing Hadoop

You can run 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.

2.14.2 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
```

2.14.3 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=#
```

2.14.4 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_*
```

2.14.5 Testing Hive

You can 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;
```

2.14.6 Testing USS

To test USS, you can add some mount-points to the USS Catalog from the Client node or the USS Namenode using the USS CLI

```
# Prepare input & output directory
hadoop fs -mkdir /tmp/test_uss_input
hadoop fs -mkdir /tmp/test_uss_output

# Copy data from /etc/passwd
hadoop fs -copyFromLocal /etc/passwd /tmp/test_uss_input

# Register local pivotal hd filesystem
# Create a sample filesystem configuration say fsconfig.xml with the following contents
# Use the appropriate value for NAMENODE_HOST

<configuration>
    <property>
        <name>uss.fs.name</name>
        <value>phdFs</value>
    </property>
    <property>
        <name>uss.fs.baseuri</name>
        <value>hdfs://NAMENODE_HOST:9000</value>
    </property>
</configuration>

$ uss fs -add fsconfig.xml

# Add mount-point for input. You can do this on any of the following hosts
# (uss-client/uss-namenode/uss-catalog)
$ uss fs -add input_mount_point phdFs /tmp/test_uss_input


# Add mount-point for output. Do this on the
# uss-client/uss-namenode/uss-catalog host
$ uss mp -add output_mount_point phdFs /tmp/test_uss_output

# List mount-points. Do this on the
# uss-client/uss-namenode/uss-catalog host
$ uss mp -list

  input_mount_point (1) > hdfs://NAMENODE_HOST:8020/tmp/test_uss_input
  output_mount_point (2) > hdfs://NAMENODE_HOST:8020/tmp/test_uss_output
```

Once you have configured the USS Catalog by adding some mount points, you can run Hadoop/Pig/Hive commands using USS URIs from any configured nodes.

```
# list contents of a mount-point
$ hadoop fs -ls uss://USS_NAMENODE_HOST:16040/input_mount_point

  Found 1 items
  -rw-r--r-- 1 user wheel 1366 2012-08-17 10:08
  hdfs://NAMENODE_HOST:8020/tmp/test_uss_input/passwd

# run word count
$ hadoop jar \
  /usr/lib/gphd/hadoop-mapreduce/hadoop-mapreduce-examples-<version>.jar \
  wordcount uss://USS_NAMENODE_HOST:16040/input_mount_point \
  uss://USS_NAMENODE_HOST:16040/output_mount_point/uss_mr_output

# run a pig script
$ cat uss.pig

  A = load 'uss://USS_NAMENODE_HOST:16040/input_mount_point';
  B = foreach A generate flatten(TOKENIZE((chararray)$0, ':')) as word;
  C = filter B by word matches '\\w+';
  D = group C by word;
  E = foreach D generate COUNT(C), group;
  STORE E into 'uss://USS_NAMENODE_HOST:16040/output_mount_point/uss_pig_output';

# Execute pig script, by adding the uss jar as an additional jar to include.
$ pig -Dpig.additional.jars=/usr/lib/gphd/uss/uss-0.4.0.jar uss.pig


# run a hive query
$ cat uss-hive.sql -- creates an external table with location pointed to by a USS URI.

DROP TABLE test_uss_external;

CREATE EXTERNAL TABLE test_uss_external (testcol1 STRING, testcol2 STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ':'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
LOCATION 'uss://USS_NAMENODE_HOST:16040/input_mount_point';

SELECT * FROM test_uss_external;



$ hive -f uss-hive.sql
```

2.15 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 it is recommended that each host would 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 it from a trusted source on the internet, copy it to your local repo directory and rerun the `createrepo` step.

2.16 High Availability (HA)

High availability is disabled by default.

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 for details.

2.16.1 Setting up a New Cluster with HA

1. Follow the instructions for Preparing the Admin Node and for fetching and editing the Cluster Configuration Files earlier in this document.

   To enable HA, you then need to make HA-specific edits to the following configuration files:
   - `clusterConfig.xml`
   - `hdfs/hdfs-site.xml`
   - `hdfs/core-site.xml`
   - `hbase/hbase-site.xml`
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</value>
</property>

<property>
  <name>dfs.ha.fencing.ssh.private-key-files</name>
  <value>/home/exampleuser/.ssh/id_rsa</value>
</property>

<property>
  <name>dfs.ha.fencing.methods</name>
  <value>shell(/bin/true)</value>
</property>

<property>
  <name>dfs.journalnode.edits.dir</name>
  <value>${journalpath}</value>
</property>

<property>
   <name>dfs.ha.automatic-failover.enabled</name>
   <value>true</value>
 </property>
```

4. Edit `hdfs/core-site.xml` as follows:
   Uncomment the following:

   ```
   <value>hdfs://${nameservices}</value>
   ```

   Comment out the following:

   ```
   <value>hdfs://${namenode}:${dfs.port}</value>
   ```

   Add following property at the end of the file:

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

5. Edit `hbase/hbase-site.xml` as follows:
   Uncomment the following:

   ```
   <value>hdfs://${nameservices}/apps/hbase/data</value>
   ```

   Comment out the following:

   ```
   <value>hdfs://${namenode}:${dfs.port}/apps/hbase/data</value>
   ```

6. Continue configuring your cluster as described earlier in this document; then deploy (see Deploying the Cluster).

2.17 Configuring GemFire XD Beta

Pivotal HD Enterprise 1.1 provides support for GemFire XD Beta. GemFire XD Beta is optional and is distributed separately from other PHD components.

GemFire XD Beta 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.*

2.17.1 Overview

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

2.17.2 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

2.17.3 Best Practices

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

2.17.4 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-x.tgz`
   If the file is a `tar`:`tar xf PRTS-1.0.x-x.tar`
3. As `gpadmin`, enable the PRTS service:

```
icm_client import -s <PATH TO EXTRACTED PRTS TAR BALL>
```

```
icm_client import -s PRTS-1.0.x-x/
```

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>`

2.17.5 Testing GemFire XD

On one of the gfxd nodes, navigate to `/usr/lib/gphd/gfxd/examples/mapreduce` and follow instructions in the README.txt file.

2.17.6 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-05.run.pivotal.io/index.jsp?topic=/com.pivotal.gemfirexd.0.5/getting_started/15-minutes.html

2.18 Installing SSL certificates

The following table contains information related to SSL certificates:

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

Port	Used by	Default Certificate Path	Default Key Path
8140	Puppet	/var/lib/puppet/ssl-icm/certs/FQDN.pem	/var/lib/puppet/ssl-icm/private_keys/FQDN.pem

3 Upgrading Pivotal HD Enterprise Using the CLI

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

3.1 Overview

Pivotal HD Enterprise only supports upgrading cluster services that were originally installed via the CLI. Services that were manually installed using tar or rpms directly on cluster hosts will not be available post-CLI upgrade and must be manually re-installed.

If you were a PCC 2.0.0 user, Hive and PXF were not automatically installed. If you manually installed those services on a cluster, they must be manually re-installed following an upgrade (see Manually re-install services).

If you were a PCC 2.0.0 user and USS was configured as one of the services in the cluster, see Upgrading if USS installed.

Complete Upgrade Syntax is provided here.

3.2 Quick Guide

The table below briefly describes the steps you need to take to upgrade a cluster; more details are provided in the following sections.

Step	Details
Prerequisites	**PADS file location**: If you are upgrading PADS, make note of the path to the extracted old PADS tar ball. **Backup Data**: We recommend that you backup any critical data before running any upgrade. **Backup Service Configuration File(s)**: Backup the configuration files of any services you will be manually re-installing, post CLI-upgrade. **Fetch original Template**: Fetch original template provided by PCC.

Step	Details
Stop Services	Stop HAWQ (if applicable). See Managing HAWQ for details. Stop all PHD services. See Managing a Cluster for details. Stop PCC (as root): `service commander stop`

Extract and upgrade PCC	Untar the new PCC package, then run(as root):
	
	`./install`
	
	change the user to gpadmin

Import and extract new stacks	Import PHD, PADS, and PHDTools For each package, run: `icm_client import -p < PATH TO EXTRACTED TAR BALL >`
CLI Self-Upgrade	`icm_client self-upgrade`
Upgrading USS Clusters	Optional: If you are upgrading a cluster that has USS configured as one of the services there are some additional pre- and post upgrade steps to take.
Upgrade PADS	`icm_client upgrade -l <CLUSTERNAME> -s pads -o < PATH TO EXTRACTED OLD ADS TAR BALL > -n < PATH TO EXTRACTED NEW ADS TAR BALL >` ⚠ This must be done first if HAWQ (PADS) is installed.
Upgrade PHD	`icm_client upgrade -l <CLUSTERNAME> -s phd`
Reinstall Manually Installed Services	Services that were manually installed on an existing cluster will not be available post-CLI upgrade and must be manually re-installed.
Post Upgrade Configuration	1. Synchronize configuration files. 2. Reconfigure the cluster

For more details instructions of the above steps, see below:

3.3 Upgrade Instructions

3.3.1 Prerequisites

- **PADS file location:** Make note of the path to the extracted old PADS tar ball; you will need this information to upgrade PADS.
- **Backup Data:** We recommend you backup any critical data before performing any backups
- **Backup Service Configuration Files:** Services that were manually installed on an existing cluster will not be available post-CLI upgrade and must be manually re-installed. Backup the configuration files for these services. See the *Pivotal HD Enterprise Stack Tool and Reference Guide* for the locations of
  these configuration files.
- **Fetch original template:** As user gpadmin, fetch original template using

  ```
  icm_client fetch-template -o ~/origTemplate
  ```

3.3.2 Upgrade Pivotal Command Center

Prerequisites:

- Stop HAWQ
- Stop all PHD services - icm_client stop -l <CLUSTER NAME>
- Stop PCC (as root): `service commander stop`

Untar the new PCC tarball, `cd` to the PCC directory, then run (as root) `./install`.

There is no need to specify that this is an upgrade; the install utility (`./install`) can detect whether it is a fresh install or an upgrade.

3.3.3 CLI Self-Upgrade

Run (as `gpadmin`) this command for the CLI to self upgrade after importing the stacks and before upgrading the stacks.

```
icm_client self-upgrade
```

3.3.4 Upgrading USS if configured

If your cluster has USS configured as one of the services there are some additional pre- and post-upgrade steps to take, as follows:
Pre-Upgrade:Before running an upgrade (PHD, PADS or both), as `gpadmin` run the following command on admin node:

```
psql gphdmgr postgres -p 10432 -c "INSERT INTO role ( service, role, gphd_version, display_name,
category, description ) VALUES ( 'uss', 'uss-admin', '2.x', 'uss-admin', 'client', 'uss admin'
)"
```

Then proceed with the upgrade.

Post-Upgrade:

After running the upgrade (PHD, PADS or both), as `gpadmin` run the following command on admin node:

```
psql gphdmgr postgres -p 10432 -c "DELETE FROM role where role='uss-admin'"
```

3.3.5 Upgrade PADS (HAWQ)

Prerequisites:

- Stop HAWQ
- Upgrade PCC
- Download and import the new PADS package
  `icm_client import -p < PATH TO EXTRACTED ADS TAR BALL >`
- Run the CLI self-upgrade
  `icm_client self-upgrade`

Upgrade:

Run (as `gpadmin`) the following command to upgrade PADS (HAWQ):

```
icm_client upgrade -l <CLUSTERNAME> -s pads -o < PATH TO EXTRACTED OLD ADS TAR
BALL > -n < PATH TO EXTRACTED NEW ADS TAR BALL >
```

This section is only applicable if you installed Pivotal ADS via the CLI, if you installed Pivotal ADS manually, refer to the *Pivotal ADS Release Notes* for upgrade instructions.

3.3.6 Upgrade Pivotal HD

Prerequisites:

- Stop services
- Upgrade PCC
- Download and import the new PHD package
  `icm_client import -p < PATH TO EXTRACTED PHD TAR BALL >`
- Run the CLI self-upgrade
  `icm_client self-upgrade`
- If the cluster is configured with HAWQ, make sure you complete upgrading Pivotal ADS (see above), before proceeding with Pivotal HD upgrade

Upgrade:

Run (as `gpadmin`) the following command to upgrade PHD:

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

3.3.7 Reinstall Manually-installed Services

1. Services that were manually installed on an existing cluster will not be available post-CLI upgrade and must be manually re-installed. Backup the configuration files for these services. See the *Pivotal HD Enterprise Stack Tool and Reference Guide* for the locations of these configuration files.
2. Perform the cluster upgrade (See Upgrade PHD and / or Upgrade PADS).
3. Manually re-install the service(s) as described in the *Pivotal HD Enterprise Stack and Tool Reference Guide*.
4. Restore the service's configuration files across the cluster

For information about manually installing services see the *Pivotal HD Enterprise Stack and Tool Reference Guide.*

3.4 Post Upgrade Configuration

3.4.1 Synchronize Configuration Files

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`.

```
icm_client fetch-template -o ~/newTemplate
```

2. Retrieve the existing configuration from database using `icm_client fetch-configuration`.

```
icm_client fetch-configuration -o ~/origConfiguration -l <CLUSTERNAME>
```

3. Identify the changes done by you using the following command:

```
diff -ruBw origTemplate/ origConfiguration/
```

Where: `origTemplate` - ICM provided template for earlier version without the user changes
`origConfiguration` - User changes on top of `origTemplate`
`newTemplate` - ICM provided template for newer version without the user changes

4. Once you have identified the changes done by you as output of step 3 `diff` command, apply those changes to the `newTemplate` you retrieved in step 1.

> ⚠ If you are re-using the `clusterConfig.xml` from `origConfiguration`, please ensure the `clusterConfig.xml` is updated with any new `servicesConfigGlobals` by checking the file in `newTemplate`

5. Upgrade or reconfigure service by specifying the cluster configuration directory as `~/newTemplate` with updated contents.

Your cluster update is now complete and you can start the cluster again.

3.5 Reconfigure Syntax

Run the reconfigure utility if you wish to upgrade the underlying stacks (Pivotal HD or ADS) in an existing cluster.

```
[gpadmin]# icm_client reconfigure --help
Usage: 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     Donot verify cluster nodes.
  -p, --nopreparehosts  Donot preparehosts as part of deploying the cluster.
  -j JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM installer binary (Ex:
                        jdk-6u41-linux-x64-rpm.bin). Ignored if -p is
                        specified
  -t, --ntp             Synchronize system clocks using NTP (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
```

3.6 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.

- Changes in Pivotal HD 1.1.1 Release
  - Changed Parameters 1.1.1
  - Sample Configuration Files for PHD 1.1.1
- Changes in Pivotal HD 1.1 Release
  - Changed Parameters 1.1
  - Sample Configuration Files for PHD 1.1

3.6.1 Changes in Pivotal HD 1.1.1 Release

Changed Parameters 1.1.1

The following parameters have been changed in PHD 1.1.1:

- hadoop-env.sh

- dfs.namenode.heapsize.mb
- dfs.datanode.heapsize.mb
- yarn.resourcemanager.heapsize.mb
- yarn.nodemanager.heapsize.mb
- dfs.datanode.failed.volumes.tolerated
- dfs.datanode.max.xcievers
- dfs.stream-buffer-size
- dfs.datanode.failed.volumes.tolerated
- yarn-env.sh
- hbase-env.sh

`hadoop-env.sh`

File Name	`hadoop-env.sh`
Value	several changes to set heapsize and other hadoop options.
Comments	Sample configuration provided.
Required	No

`dfs.namenode.heapsize.mb`

File Name	`clusterConfig.xml`
Value	`2048`
Comments	Max and Min heapsize setting for namenode process
Required	Yes

`dfs.datanode.heapsize.mb`

File Name	`clusterConfig.xml`
Value	`2048`
Comments	Max and Min heapsize setting for datanode process
Required	No

`yarn.resourcemanager.heapsize.mb`

File Name	`clusterConfig.xml`
Value	`2048`
Comments	Max and Min heapsize setting for resourcemanager process
Required	Yes

`yarn.nodemanager.heapsize.mb`

File Name	`clusterConfig.xml`
Value	`2048`
Comments	Min heapsize setting for nodemanager process
Required	Yes

`dfs.datanode.failed.volumes.tolerated`

File Name	`clusterConfig.xml`
Value	0
Comments	The number of volumes that are allowed to fail before a datanode stops offering service. By default any volume failure will cause a datanode to shutdown
Required	Yes

`dfs.datanode.max.xcievers`

File Name	`hdfs-site.xml`
Value	Removed
Comments	This property has been removed.
Required	No

`dfs.stream-buffer-size`

File Name	`hdfs-site.xml`
Value	131072
Comments	

	The size of buffer to stream files. The size of this buffer should probably be a multiple of hardware page size (4096 on Intel x86), and it determines how much data is buffered during read and write operations.
Required	Yes

`dfs.datanode.failed.volumes.tolerated`

File Name	`hive-site.xml`
Value	`{dfs.datanode.failed.volumes.tolerated}`
Comments	The number of volumes that are allowed to fail before a datanode stops offering service. By default any volume failure will cause a datanode to shutdown.
Required	Yes

`yarn-env.sh`

File Name	`yarn-env.sh`
Value	Several changes to set heapsize and other options.
Comments	Sample Configuration file provided
Required	

`hbase-env.sh`

File Name	`hadoop-env.sh`
Value	
Comments	Added: `export HBASE_HEAPSIZE=${hbase. heapsize.mb}`
Required	

Sample Configuration Files for PHD 1.1.1

Here is a sample `hadoop-env.sh` :

```
# The java implementation to use.  Required.
export JAVA_HOME=${cluster_java_home}

# The maximum amount of heap to use, in MB. Default is 1000.
export HADOOP_HEAPSIZE=1024
export HADOOP_NAMENODE_HEAPSIZE=${dfs.namenode.heapsize.mb}
export HADOOP_DATANODE_HEAPSIZE=${dfs.datanode.heapsize.mb}

# Extra Java runtime options. Empty by default.
export HADOOP_OPTS="-Djava.net.preferIPv4Stack=true ${HADOOP_OPTS}"

# Extra ssh options.  Empty by default.
export HADOOP_SSH_OPTS="-o ConnectTimeout=5 -o SendEnv=HADOOP_CONF_DIR"

# Set Hadoop-specific environment variables here.

# Command specific options appended to HADOOP_OPTS when specified
export HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote -Xms${dfs.namenode.heapsize.mb}m
-Xmx${dfs.namenode.heapsize.mb}m -Dhadoop.security.logger=INFO,DRFAS
-Dhdfs.audit.logger=INFO,DRFAAUDIT -XX:ParallelGCThreads=8 -XX:+UseParNewGC
-XX:+UseConcMarkSweepGC -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps
-Xloggc:${HADOOP_LOG_DIR}/hadoop-hdfs-namenode-`date +'%Y%m%d%H%M'`.gclog
-XX:ErrorFile=${HADOOP_LOG_DIR}/hs_err_pid%p.log $HADOOP_NAMENODE_OPTS"

export HADOOP_SECONDARYNAMENODE_OPTS="-Dcom.sun.management.jmxremote
-Xms${dfs.namenode.heapsize.mb}m -Xmx${dfs.namenode.heapsize.mb}m
-Dhadoop.security.logger=INFO,DRFAS -Dhdfs.audit.logger=INFO,DRFAAUDIT -XX:ParallelGCThreads=8
-XX:+UseParNewGC -XX:+UseConcMarkSweepGC -verbose:gc -XX:+HeapDumpOnOutOfMemoryError
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps
-Xloggc:${HADOOP_LOG_DIR}/hadoop-hdfs-secondary-namenode-`date +'%Y%m%d%H%M'`.gclog
-XX:ErrorFile=${HADOOP_LOG_DIR}/hs_err_pid%p.log $HADOOP_SECONDARYNAMENODE_OPTS"

export HADOOP_DATANODE_OPTS="-Dcom.sun.management.jmxremote -Xms${dfs.datanode.heapsize.mb}m
-Xmx${dfs.datanode.heapsize.mb}m -Dhadoop.security.logger=ERROR,DRFAS $HADOOP_DATANODE_OPTS"

export HADOOP_BALANCER_OPTS="-Dcom.sun.management.jmxremote -server -Xmx${HADOOP_HEAPSIZE}m
$HADOOP_BALANCER_OPTS"

export HADOOP_JOBTRACKER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_JOBTRACKER_OPTS"

# export HADOOP_TASKTRACKER_OPTS=
# The following applies to multiple commands (fs, dfs, fsck, distcp etc)
# export HADOOP_CLIENT_OPTS

# The following applies to multiple commands (fs, dfs, fsck, distcp etc)
export HADOOP_CLIENT_OPTS="-Xmx${HADOOP_HEAPSIZE}m $HADOOP_CLIENT_OPTS"

# GPHD variables
export GPHD_HOME=/usr/lib/gphd
export GPHD_CONF=/etc/gphd

# PXF conf directory
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_CONF/pxf/conf:\

# Required for PXF with HDFS
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_HOME/pxf/pxf.jar:\
$GPHD_HOME/publicstage:\
$GPHD_HOME/pxf/avro-1.5.4.jar:\
$GPHD_HOME/pxf/avro-mapred-1.5.4.jar:\
$GPHD_HOME/gfxd/lib/sqlfire.jar:\
# Required only for PXF with HBase
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_HOME/zookeeper/zookeeper.jar:\
$GPHD_HOME/hbase/hbase.jar:\
$GPHD_CONF/hbase/conf:\

# Required only for PXF with HDFS & Hive
export HIVELIB_HOME=$GPHD_HOME/hive/lib
export HIVE_CONF=$GPHD_CONF/hive/conf
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$HIVELIB_HOME/hive-service-0.11.0-gphd-2.1.1.0.jar:\
$HIVELIB_HOME/libthrift-0.9.0.jar:\
$HIVELIB_HOME/hive-metastore-0.11.0-gphd-2.1.1.0.jar:\
$HIVELIB_HOME/libfb303-0.9.0.jar:\
$HIVELIB_HOME/hive-common-0.11.0-gphd-2.1.1.0.jar:\
$HIVELIB_HOME/hive-exec-0.11.0-gphd-2.1.1.0.jar:\
$HIVELIB_HOME/postgresql-jdbc.jar:\
$HIVE_CONF:\

# Required only for USS
export USS_HOME=$GPHD_HOME/uss
export USS_CONF=$GPHD_CONF/uss/conf
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$USS_HOME/*:\
$USS_CONF:

HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_HOME/sm-plugins/*:

export HADOOP_CLASSPATH
```

Here is a sample `yarn-env.sh`:

```
# User for YARN daemons
export HADOOP_YARN_USER=${HADOOP_YARN_USER:-yarn}

# resolve links - $0 may be a softlink
export YARN_CONF_DIR="${YARN_CONF_DIR:-$HADOOP_YARN_HOME/conf}"

# some Java parameters
export JAVA_HOME=${cluster_java_home}
if [ "$JAVA_HOME" != "" ]; then
  #echo "run java in $JAVA_HOME"
  JAVA_HOME=$JAVA_HOME
fi

if [ "$JAVA_HOME" = "" ]; then
  echo "Error: JAVA_HOME is not set."
  exit 1
fi

JAVA=$JAVA_HOME/bin/java
JAVA_HEAP_MAX=-Xmx1000m

# check envvars which might override default args
if [ "$YARN_HEAPSIZE" != "" ]; then
```

```
  #echo "run with heapsize $YARN_HEAPSIZE"
  JAVA_HEAP_MAX="-Xmx""$YARN_HEAPSIZE""m"
  #echo $JAVA_HEAP_MAX
fi

# so that filenames w/ spaces are handled correctly in loops below
IFS=

# default log directory & file
if [ "$YARN_LOG_DIR" = "" ]; then
  YARN_LOG_DIR="$HADOOP_YARN_HOME/logs"
fi
if [ "$YARN_LOGFILE" = "" ]; then
  YARN_LOGFILE='yarn.log'
fi

# default policy file for service-level authorization
if [ "$YARN_POLICYFILE" = "" ]; then
  YARN_POLICYFILE="hadoop-policy.xml"
fi

# restore ordinary behaviour
unset IFS
YARN_OPTS="$YARN_OPTS -Dhadoop.log.dir=$YARN_LOG_DIR"
YARN_OPTS="$YARN_OPTS -Dyarn.log.dir=$YARN_LOG_DIR"
YARN_OPTS="$YARN_OPTS -Dhadoop.log.file=$YARN_LOGFILE"
YARN_OPTS="$YARN_OPTS -Dyarn.log.file=$YARN_LOGFILE"
YARN_OPTS="$YARN_OPTS -Dyarn.home.dir=$YARN_COMMON_HOME"
YARN_OPTS="$YARN_OPTS -Dyarn.id.str=$YARN_IDENT_STRING"
YARN_OPTS="$YARN_OPTS -Dhadoop.root.logger=${YARN_ROOT_LOGGER:-INFO,console}"
YARN_OPTS="$YARN_OPTS -Dyarn.root.logger=${YARN_ROOT_LOGGER:-INFO,console}"
if [ "x$JAVA_LIBRARY_PATH" != "x" ]; then
  YARN_OPTS="$YARN_OPTS -Djava.library.path=$JAVA_LIBRARY_PATH"
fi
YARN_OPTS="$YARN_OPTS -Dyarn.policy.file=$YARN_POLICYFILE"

# Yarn daemon heap sizes
export YARN_RESOURCEMANAGER_HEAPSIZE=${yarn.resourcemanager.heapsize.mb}
export YARN_NODEMANAGER_HEAPSIZE=${yarn.nodemanager.heapsize.mb}

# common jvm settings for resource manager and node managers
YARN_OPTS="$YARN_OPTS -server -Djava.net.preferIPv4Stack=true -XX:+UseParNewGC
-XX:+UseConcMarkSweepGC -XX:+HeapDumpOnOutOfMemoryError -XX:+PrintGCDetails
-XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:ErrorFile=${YARN_LOG_DIR}/hs_err_pid%p.log"

# yarn resource manager related jvm settings
YARN_RESOURCEMANAGER_OPTS="$YARN_RESOURCEMANAGER_OPTS
-Xloggc:${YARN_LOG_DIR}/hadoop-yarn-resourcemanager-`date +'%Y%m%d%H%M'`.gclog"

# yarn node manager related jvm settings
YARN_NODEMANAGER_OPTS="$YARN_NODEMANAGER_OPTS
-Xloggc:${YARN_LOG_DIR}/hadoop-yarn-nodemanager-`date +'%Y%m%d%H%M'`.gclog"
```

3.6.2 Changes in Pivotal HD 1.1 Release

The following files were add/removed in PHD 1.1:

File	Description	Req'd
`uss-env.sh`	This is a new config file added to the USS folder in `clusterConfig` template	Yes
`pxf-profiles.xml`	This is a new config file added to the `gpxf` folder in `clusterConfig` template	Yes
`capacity-scheduler.xml`	This is a new config file added to the yarn folder in `clusterConfig` template.	Yes

Changed Parameters 1.1

The following parameters were changed in PHD 1.1

- cluster_java_home
- uss-admin
- JAVA_HOME
- HBASE_CLASSPATH
- fs.uss.enabled
- mapred.outdir.resolverClass
- HADOOP_CLASSPATH
- hive.hwi.war.file
- yarn.application.classpath
- gpxf

`cluster_java_home`

File Name	`clusterConfig.xml`
Value	`/usr/java/default`
Comments	`JAVA_HOME` defaults to this. Update if you have java installed elsewhere
Required	Yes

`uss-admin`

File Name	`clusterConfig.xml`
Value	`Removed`
Comments	This property has been removed.
Required	No

`JAVA_HOME`

File Name	`hbase-env.sh, hadoop-env.sh,mapred-env.sh, yarn-env.sh`
Value	`${cluster_java_home}`
Comments	`JAVA_HOME` variable is now parameterized
Required	Yes

`HBASE_CLASSPATH`

File Name	`hbase-env.sh`
Value	`${HBASE_CLASSPATH}:\ $GPHD_ROOT/pxf/pxf.jar`
Comments	`gpxf` folder and jar renamed to `pxf`.
Required	Yes

`fs.uss.enabled`

File Name	`core-site.xml`
Value	Removed
Comments	This property has been removed.
Required	No

`mapred.outdir.resolverClass`

File Name	`core-site.xml`
Value	Removed
Comments	This property has been removed.

Required	No

HADOOP_CLASSPATH

File Name	`hadoop-env.sh`
Value	Several changes to hadoop classpath
Comments	Sample configuration provided.
Required	No

hive.hwi.war.file

File Name	`hive-site.xml`
Value	`/usr/lib/gphd/hive/lib/hive-hwi.war`
Comments	
Required	Yes

yarn.application.classpath

File Name	`yarn-site.xml`
Value	`$HADOOP_CONF_DIR, $HADOOP_COMMON_HOME/*, $HADOOP_COMMON_HOME/lib/*, $HADOOP_HDFS_HOME/*, $HADOOP_HDFS_HOME/lib/*, $HADOOP_MAPRED_HOME/*, $HADOOP_MAPRED_HOME/lib/*, $HADOOP_YARN_HOME/*, $HADOOP_YARN_HOME/lib/*, $USS_HOME/*,$USS_CONF`
Comments	USS-related configs added to this property.
Required	Yes

gpxf

File Name	`hadoop-env.sh`
Value	`pxf`

Comments	Change all occurrence of `gpxf` to `pxf`
	
	Example
	
	`# Required for PXF with HDFS`
	`HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\`
	`$GPHD_HOME/pxf/pxf.jar:\`
	`$GPHD_HOME/publicstage:\`
	`$GPHD_HOME/pxf/avro-1.5.4.jar:\`
	`$GPHD_HOME/pxf/avro-mapred-1.5.4.jar:\`

Required	

Sample Configuration Files for PHD 1.1

Here is a sample `hadoop-env.sh` :

```
# Set Hadoop-specific environment variables here.
# The java implementation to use.  Required.
export JAVA_HOME=${cluster_java_home}
# Command specific options appended to HADOOP_OPTS when specified
export HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_NAMENODE_OPTS"
export HADOOP_SECONDARYNAMENODE_OPTS="-Dcom.sun.management.jmxremote
$HADOOP_SECONDARYNAMENODE_OPTS"
export HADOOP_DATANODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_DATANODE_OPTS"
export HADOOP_BALANCER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_BALANCER_OPTS"
export HADOOP_JOBTRACKER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_JOBTRACKER_OPTS"
# export HADOOP_TASKTRACKER_OPTS=
# The following applies to multiple commands (fs, dfs, fsck, distcp etc)
# export HADOOP_CLIENT_OPTS
# Extra ssh options.  Empty by default.
# export HADOOP_SSH_OPTS="-o ConnectTimeout=1 -o SendEnv=HADOOP_CONF_DIR"
# Where log files are stored.  $HADOOP_HOME/logs by default.
export HADOOP_LOG_DIR=$HADOOP_HOME/logs
# GPHD variables
export GPHD_HOME=/usr/lib/gphd
export GPHD_CONF=/etc/gphd

# PXF conf directory
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_CONF/pxf/conf:

# Required for PXF with HDFS
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_HOME/pxf/pxf.jar:\
$GPHD_HOME/publicstage:\
$GPHD_HOME/pxf/avro-1.5.4.jar:\
$GPHD_HOME/pxf/avro-mapred-1.5.4.jar:\
$GPHD_HOME/gfxd/lib/sqlfire.jar:

# Required only for PXF with HBase
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_HOME/zookeeper/zookeeper.jar:\
$GPHD_HOME/hbase/hbase.jar:\
$GPHD_CONF/hbase/conf:

# Required only for PXF with HDFS & Hive
export HIVELIB_HOME=$GPHD_HOME/hive/lib
export HIVE_CONF=$GPHD_CONF/hive/conf
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$HIVELIB_HOME/hive-service-0.11.0-gphd-2.1.0.0.jar:\
$HIVELIB_HOME/libthrift-0.9.0.jar:\
$HIVELIB_HOME/hive-metastore-0.11.0-gphd-2.1.0.0.jar:\
$HIVELIB_HOME/libfb303-0.9.0.jar:\
$HIVELIB_HOME/hive-common-0.11.0-gphd-2.1.0.0.jar:\
$HIVELIB_HOME/hive-exec-0.11.0-gphd-2.1.0.0.jar:\
$HIVELIB_HOME/postgresql-jdbc.jar:\
$HIVE_CONF:

# Required only for USS
export USS_HOME=$GPHD_HOME/uss
export USS_CONF=$GPHD_CONF/uss/conf
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$USS_HOME/*:\
$USS_CONF:

HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
$GPHD_HOME/sm-plugins/*:

export HADOOP_CLASSPATH
```

4 Administering Pivotal HD Enterprise Using the CLI

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

4.1 Managing a Cluster

4.1.1 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, mahout and uss.
`-r`	Starts only the specified role on a specific set of hosts. Hosts can be specified using the -o option.

Option	Description
`-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.

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
```

4.1.2 Stopping a Cluster

You can use the *stop* option 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, mahout, and uss.
`-r`	Stops the specified role on a specific set of hosts. Hosts can be specified using the -o option.

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
```

4.1.3 Restarting a Cluster

You can use the `-restart` option 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
```

4.1.4 Reconfiguring a Cluster

Run the `reconfigure` command to update specific configuration for an existing cluster. Some cluster specific configuration cannot be updated:

!Topology of a cluster (host to role mapping) are not allowed. For example: changing the NameNode to a different node or adding new set of datanodes to a cluster

!Properties derived 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 cannot be changed.

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`

```
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     Donot verify cluster nodes.
  -p, --nopreparehosts  Donot preparehosts as part of deploying the cluster.
  -j JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM installer binary (Ex:
                        jdk-6u41-linux-x64-rpm.bin). Ignored if -p is
                        specified
  -t, --ntp             Synchronize system clocks using NTP (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
```

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.

4.1.5 Add / Remove Services

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

- Edit the `clusterConfig.xml` file to add / remove services from the service list in 'services' tag
- Edit `hostRoleMapping` section to add/remove hosts for the specific services configured
- Edit the `servicesConfigGlobals` if required for the specific service added
- Follow the steps for Reconfiguring a Cluster.

- Like in a new deploy 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 removed when the Cluster is uninstalled

To add Hbase or HAWQ

Since the slave nodes like Hbase region servers or HAWQ segments have to be co-located with datanodes, if you plan to add new nodes to your cluster, you will first have to expand the existing cluster using `add-slaves` command and then use reconfigure to add Hbase or HAWQ service. If you plan to just reuse the nodes in the existing cluster then you can directly use reconfigure to add the new service.

The steps to add new hosts to the cluster:

1. Prepare the new hosts using the `icm_client preparehosts` command.
2. Add the new hosts that will serve as slave roles (like Hbase region server or HAWQ segment servers) to the cluster using add-slaves. Any new node that will be added as a master role need not be added in this step
3. Add the new service like Hbase or HAWQ using the reconfigure step mentioned above.
4. Note: To install Hive, you need not run the add-slaves as all the hive roles are considered master roles. You can directly use the reconfigure to add Hive service.

!Please note there is a limitation that you cannot add one service and remove another at the same time. They will have to be two separate steps but you can add multiple services OR remove multiple services at the same time.

4.1.6 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
```

4.1.7 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
```

4.1.8 Expanding a Cluster

!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.)

Run the `add-slaves` command to add additional slave hosts to an existing cluster. All the slave roles for the existing cluster services will be installed on the new cluster hosts.

```
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
  -s, --noscanhosts     Donot verify cluster nodes.
  -j JAVAHOME, --java_home=JAVAHOME
                        JAVA_HOME path to verify on cluster nodes
  -p, --nopreparehosts  Donot preparehosts as part of deploying the cluster.
  -k JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM installer binary (Ex:
                        jdk-6u41-linux-x64-rpm.bin). Ignored if -p is
                        specified
  -t, --ntp             Synchronize system clocks using NTP (requires external
                        network access) for the newly added nodes. 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 of the newly added slave hosts.

```
icm_client start -l CLUSTERNAME -r datanode -o hostfile
icm_client start -l CLUSTERNAME -r yarn-nodemanager -o hostfile
```

!If HBase is configured, start hbase-regionservers as well.

!Don't expect data blocks to be distributed to the newly added slave nodes immediately.

4.1.9 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
  -s, --noscanhosts     Donot verify cluster nodes.
  -j JAVAHOME, --java_home=JAVAHOME
                        JAVA_HOME path to verify on cluster nodes
  -p, --nopreparehosts  Donot preparehosts as part of deploying the cluster.
  -k JDKPATH, --java=JDKPATH
                        Location of Sun Java JDK RPM installer binary (Ex:
                        jdk-6u41-linux-x64-rpm.bin). Ignored if -p is
                        specified
  -t, --ntp             Synchronize system clocks using NTP (requires external
                        network access) for the newly added nodes. 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 remove-slaves -l CLUSTERNAME -f hostfile
```

4.1.10 Enabling High Availability on a Cluster

High availability is disabled by default.

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

To enable HA for an existing cluster, see below.

1. Download and import the latest version of Pivotal Command Center (PCC) (see Installing Pivotal HD Enterprise Using the CLI for details)
   Note: PCC 2.1 is the first version to support HA.
2. Reconfigure your 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. Fetch the new template configuration:
      `icm_client fetch-template -o ~/ClusterConfigDir`
   4. Merge the HA-related configuration changes into your existing cluster configuration.
      See the High Availability section of Installing Pivotal HD Enterprise Using the CLI for details of the HA-specific information you need to add to the configuration files.
   5. Reconfigure the cluster:
      `icm_client reconfigure -l CLUSTERNAME -c LOCALDIR`

3. 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.
   1. Login to host configured as `hive-metastore`.
   2. Display the current NameNode and hdfspath for hive warehouse directory:
      `/usr/lib/gphd/hive/bin/metatool -listFSRoot`
   3. 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.

You can now start the entire cluster with all configured services running.

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
```

4.1.11 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 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 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.

4.1.12 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.

!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
```

Note: 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.

4.2 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 1.0 Installation Guide* and the *Pivotal ADS 1.0 Administrator Guide*.

4.2.1 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.

!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

4.2.2 Starting HAWQ

Run the `start` command to start up the HAWQ master and all the segments hosts including the Postgres database. This is implicitly done as part of the HAWQ Initialization.

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

4.2.3 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
```

4.2.4 Modifying HAWQ User Configuration

If you are using PCC, 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 to 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;"
```

4.3 Managing Roles and Hosts

Pivotal HD supports starting or stopping entire clusters or individual roles on a selected hosts. If you wish to start and stop the roles manually you can follow these steps:

You have the following options when managing cluster and individual roles:

-
    - Managing locally
    - Managing from the Admin Node

4.3.1 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
```

4.3.2 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' containing 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}`
USS Namenode	`sudo /etc/init.d/uss-namenode {start|stop|status|restart}`
Quorum Journal node	`sudo /etc/init.d/hadoop-hdfs-journalnode {start|stop|status|restart}`

4.4 Pivotal HD Services Reference

4.4.1 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
		mapred:hadoop	777

Service	Directory	Owner	Permissions
	mapreduce.jobhistory.intermediate-done-dir (/user/history/done)		
	mapreduce.jobhistory.done-dir (/user/history/done)	*mapred:hadoop*	777
	yarn.nodemanager.remote-app-log-dir	*mapred:hadoop*	755
HBase	*hbase directory (/apps/hbase/data)*	*hdfs:hadoop*	775
HAWQ	*hawq directory (/hawq_data)*	*hdfs:hadoop*	755

4.4.2 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
DataLoader	dladmin	dladmin	Yes

Note: USS does not use any Linux users. USS creates and uses a `usscatalog` database role for managing the USS catalog postgres database.

4.4.3 Pivotal HD Ports

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

Service	Port
ssh	22
NameNode	8020 (Apache 9000)

Service	Port
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
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

5 Pivotal HD Enterprise FAQ

5.1 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.

5.2 Can I modify the topology (host to role mapping) of the cluster after the initial install?

No, you cannot change the topology.

5.3 How do I reformat the namenode?

Warning: These steps will erase all data on 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. Run: "hadoop namenode format -force" on the name node.

5.4 Certain services such as hadoop-hdfs-namenode or hadoop-hdfs-datanode do not come up when I perform "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

5.5 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.

5.6 What is the allowed time difference amongst the cluster nodes v/s 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.

5.7 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.

5.8 Does PCC support hostname both in IP address and FQDN format?

No, only FQDN format is currently supported.

5.9 Can a node be shared between different clusters?

No, nodes cannot be shared between clusters

5.10 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. Please uninstall Puppet and install the one shipped with the product using the *icm_client preparehosts* command.

5.11 How do I clean up the nodes if a cluster deployment failed?

Uninstall the cluster using the *icm_client* command and follow instructions for deploying the cluster again.

5.12 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.

5.13 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.

5.14 Can I upgrade Pivotal Command Center/HDManager while the clusters are functioning?

Yes you can. Upgrading the Admin node will not interfere with any of the clusters.

5.15 How do I change the port used by Pivotal HD?

1. Log onto the machine as root.
2. Stop 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 Command Center again

```
service commander start
```

6 Pivotal HD Enterprise Troubleshooting

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

6.1 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:

6.2 Pivotal HD Installation

All installation errors will be logged under:
`/var/log/gphd/gphdmgr/installer.log`

6.3 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 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
```

6.4 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.

6.5 Services Start

Check for the corresponding log file under `/var/log/gphd/` directory.

For example, if the namenode doesn't start, please look at the `/var/log/gphd/hadoop/hadoop-hdfs-namenode-hostname.log` file for details.

6.6 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.

6.7 Upgrade/Reconfigure Errors

6.7.1 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>
```

6.7.2 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 /usr/lib/gphd directory. Point the hadoop-hdfs symlink to the newer hadoop-hdfs version.

If 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 please ensure to maintain the order in the comma separated list.

If you see "ERROR:Fetching hadoop rpm name on namenode: <*host*> failed" error, it is most likely a case where the cluster was being upgraded from 1.0.0 to 1.0.2 and there was an error during upgrade.
Workaround: Run `yum install hadoop-2.0.2_alpha_gphd_2_0_1_0-14.x86_64` on the namenode and retry upgrade.

If you are upgrading a cluster with hbase, hive or pxf configured as a service you must manually reinstall those services. See Upgrading Pivotal HD Enterprise Using the CLI for details.

6.8 HA-related Errors

If the cluster fails to start with HA enabled:

- Check status of 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 make sure to 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.

6.9 Other Errors

6.9.1 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 NFS mount point on all the cluster nodes

6.9.2 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>
```

6.9.3 Installation Fails due to Directory Permissions

Please check if the umask is set to 0022. If not please set the umask in the .bashrc as "umask 0022" and retry the PCC installation.

6.9.4 Deployment Fails due to Problems with yum Repository

Please verify that the admin node is reachable from the agent node.
If you have configured proxy servers, please refer the section titled "Working with Proxy servers" in the troubleshooting section on the work arounds.

6.9.5 Installation Fails due to Problems with the SSL certificate

Check if `dnsdomainname` returns empty value. If yes, you need to ensure that the `dnsdomainname` returns the correct domain.

6.9.6 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`)

And retry the deployment

6.9.7 Puppet certificate failure

Follow the instructions in the "Handling Puppet SSL errors" of Troubleshooting

6.9.8 Package Bundle Not Found

If you sudo in to 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 password and if you still see the above issue, check if /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` and save
`[]# source ~/.bashrc`

6.9.9 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.

6.9.10 Working with Proxy Servers

It might sometimes be required for all outgoing http traffic to use a HTTP proxy. PCC installer might sometimes pull rpms from external repos like 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` 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 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 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/}
```

6.9.11 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.

6.9.12 Resolving postgres port conflict issue

If you face a postgres port conflict or wish to change the default postgres port, please 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
   ```

6.9.13 Resolving HTTP port conflict

Check the FAQ section: How do I change the port used by Pivotal HD?

6.9.14 Seeing 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) which are substrings of other hostnames then this issue seems to occur.

For example: host1.emc.com, host11.emc.com

This error can be ignored for now as the actual deployment still goes through.

6.9.15 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)

6.9.16 HAWQ initialization failing

Make sure your cluster is up and running with the hadoop services prior to running 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. Please refer to the HAWQ guide to configure/modify the system memory requirements.

6.9.17 Installing HAWQ on dirty cluster nodes previously configured with HAWQ

If you wish the deploy or initialize HAWQ on

a) Cluster which had an older uninstalled hawq cluster, or

b) Cluster that failed in its attempts to initialize hawq, you will need to do the following before Intializing HAWQ with the new cluster nodes.

`HAWQ_Hosts.txt` contains all the HAWQ hosts that you want to clean up.

You will need to run the below 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.

6.9.18 Errors Related to VM Memory

If you are planning to deploy a HAWQ cluster on VMs with memory lower than the optimized/recommended requirements, you may encounter *Could not create the Java virtual machine* type errors. In these cases you can reconfigure memory usage, as follows:

- Remove the entry `vm.overcommit_memory = 2` from `/usr/lib/gphd/gphdmgr/hawq_sys_config/sysctl.conf` prior to running the prepare HAWQ utility.
- In the `clusterConfig.xml`, update `<hawq.segment.directory>` to include only one segment directory entry (instead of the default 4 segments).