Pivotal™

Pivotal HD Enterprise 1.0
Installation and Administrator Guide

Rev: A07

Use of Open Source

This product may be distributed with open source code, licensed to you in accordance with the applicable open source license. If you would like a copy of any such source code, EMC will provide a copy of the source code that is required to be made available in accordance with the applicable open source license. EMC may charge reasonable shipping and handling charges for such distribution. Please direct requests in writing to EMC Legal, 176 South St., Hopkinton, MA 01748, ATTN: Open Source Program Office.

Date: 9/26/13

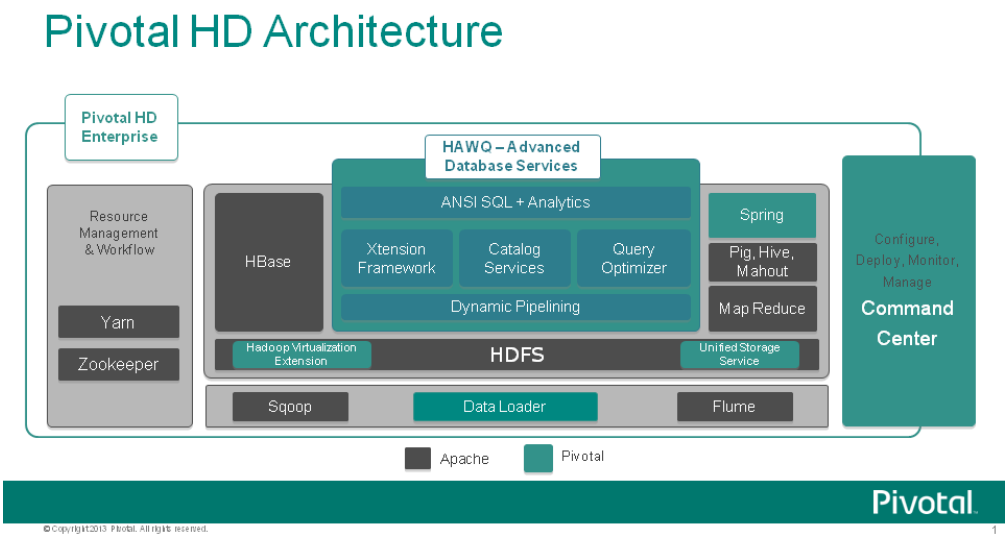
Overview of Pivotal HD Enterprise

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
  - Pig
  - Hive
  - HBase
  - Mahout
  - Zookeeper
  - Flume
  - Sqoop
  - HVE
  - Oozie

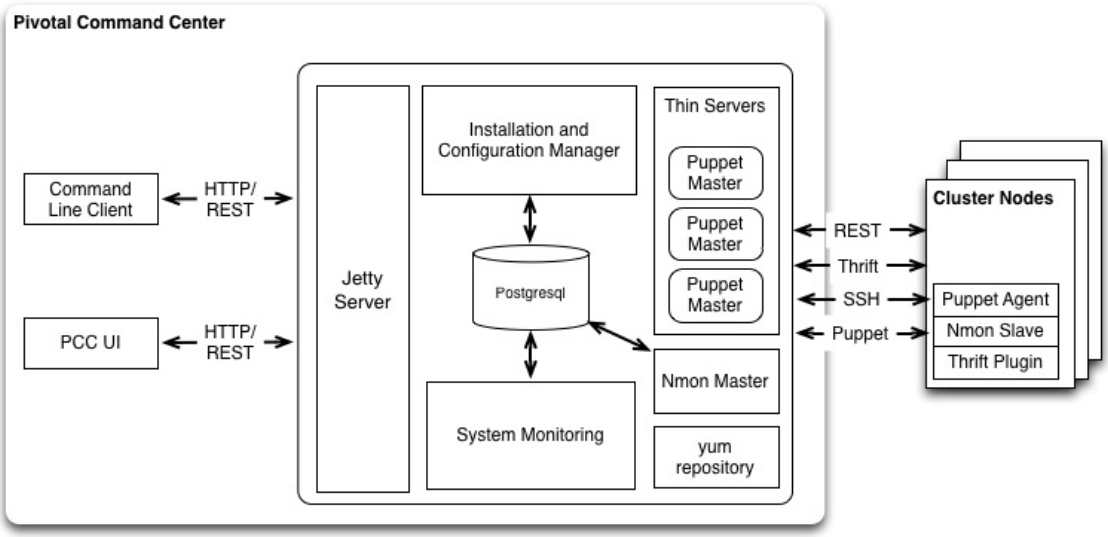
Pivotal HD Enterprise enriches the Apache stack distribution by providing the following:

- **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 monitoring & management of Pivotal HD environment. With the help of PCC, system administrators can determine if the PHD cluster is running efficiently, quickly diagnose functional or performance issues, and performs cluster management tasks when required.



Pivotal Command Center (PCC) includes a CLI (command line interface) and a GUI. You deploy and configure most of the Hadoop services as well as HAWQ, PXF and USS, using the CLI. You can start and stop the clusters using either the CLI or the GUI. PCC stores the metadata for Hadoop cluster nodes and services, the cluster configuration and the system metrics in a PostgreSQL database.

See Deployment Options.



About Pivotal, Inc.

Greenplum is currently transitioning to a new corporate identity (Pivotal, Inc.). We estimate that this transition will be completed in 2013. During this transition, there will be some legacy instances of our former corporate identity (Greenplum) appearing in our products and documentation.

If you have any questions or concerns, please do not hesitate to contact us through our web site:

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

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.

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

Pivotal HD 1.0.3 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.

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.

Notes

- HBase requires that you have installed HDFS, YARN, and Zookeeper.
- Pivotal HD Enterprise installs ZooKeeper if you have not installed it.
- Pivotal HD Enterprise only supports Distributed Mode (Not Standalone Mode)
- Pivotal HD Enterprise 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
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 using PCC's 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 HAWQ.

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

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.

Deployment Options

Pivotal HD 1.0.3 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.

Note that if you choose to deploy an MR1 based cluster, you can only use the components provided in the MR1 package you downloaded; you cannot combine a stack component from an MR1 package with one from an MR2 package.

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	✓	✓	✓
Hadoop MR1:	MapReduce 1.0.3		✓	✓
Pig		✓	✓	✓
Hive		✓	✓	✓
HBase		✓	✓	✓
Mahout		✓	✓	✓
Zookeeper		✓	✓	✓
Flume		✓	✓	✓
Hcatalog			✓	✓
Sqoop			✓	✓
HVE			✓	✓
Oozie				✓
Advanced Database	HAWQ	✓	✓	✓

Services:	PXF	✔	✔	✔
PHD Tools	USS	✔	✔	✔
	DataLoader		✔	✔

Installation Instructions:

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

- Pivotal Command Center: *Pivotal HD Enterprise Installation and Adminstrator Guide* (this document)
- Hadoop MR2 (via CLI): *Pivotal HD Enterprise Installation and Adminstrator Guide* (this document)
- Hadoop MR2 and MR1 manual installs: *Pivotal HD Enterprise 1.0 Stack and Tool Reference Guide*.
- USS: *Pivotal HD Enterprise 1.0 Stack and Tool Reference Guide*.
- Pivotal DataLoader: *Pivotal HD 2.0 Installation and Administrator Guide*.

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 Pivotal HD 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

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.

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 1.6. If not, download the appropriate version from Oracle. Note: Oracle JDK 1.7 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.
	Verify iptables is turned off (as root) # chkconfig iptables off # service iptables stop # service iptables status iptables: Firewall is not running.
	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 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
	Enable Pivotal HD Service (as gpadmin) # icm_client import -p < ***PATH TO EXTRACTED PHD TAR BALL*** >
	Enable HAWQ and PXF services (as gpadmin) # icm_client import -p < ***PATH TO EXTRACTED ADS TAR BALL*** >
	Enable USS services Optional (as gpadmin) # icm_client import -p < ***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.
	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.
Prepare the HAWQ Nodes This is required only if you plan to install HAWQ See Preparing the Cluster Nodes for more details	**Create the HostFile for HAWQ** (as gpadmin) Create a hostfile (*HAWQ_Hosts.txt*) that contains the hostnames of all your HAWQ nodes (HAWQ master, standby master, and segment nodes) For example: [gpadmin] cat *HAWQ_Hosts*.txt host3.pivotal.com host4.pivotal.com host5.pivotal.com
	Prepare the HAWQ nodes (as gpadmin) # icm_client prepare-hawq-hosts --hostfile=./HAWQ_Hosts.txt -g /usr/lib/gphd/gphdmgr/hawq_sys_config/

	Verify the Cluster Nodes # icm_client scanhosts -f ./HostFile.txt
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 user 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

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 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
```

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
```

Important: 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 1.6.

> Note: Oracle JDK 1.6 has been tested with PHD 1.6. Oracle JDK 1.7 is optional, but not fully tested at this time. Customers may use JDK 1.7, 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.

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

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 1.6 Package - For example: jdk-6u43-linux-x64-rpm.bin (you can download this from the Oracle site at http://www.oracle.com/technetwork/java/javase/downloads/index.html

  Note that Oracle JDK 1.6 has been tested with PHD 1.0.x. Oracle JDK 1.7 is optional, but not fully tested at this time. Customers may use JDK 1.7, 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.

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.

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
```

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: 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

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, 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 you may have to edit the HAWQ configuration file, see Configuring HAWQ.

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
```

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.

Note: Once you've made your changes, we recommend you check that your xml is well-formed using the *xmlwf* command.

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.

DOC Note: we will be adding a sample config file once we have a completed copy

The *clusterConfig.xml* contains a default Cluster Configuration template. The following is an example of the directory structure in a *clusterConfig.xml* file:

```
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 *cluste rConfig.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.

! **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.

! *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

Note: 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.

Note: 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>
```

Configuring HAWQ

You can find the HAWQ system configuration 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.

! **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

Configuring USS

The USS configuration files are located in *uss/*.

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.

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.

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 PCC's 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.

Preparing the Cluster Nodes for HAWQ

This command has to be run only if you plan to install HAWQ. The *prepare-hawq-hosts* comman d 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
```

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:

- */tmp/ScanHost.XXX.log*

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

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 >= 1.6 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.
3. 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>
   ```

4. 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>
```

5. Once the file is imported, you can just use the name of the file in the option "-j" during deploy.

! 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
```

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.

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

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.

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).

Please refer to the services scripts section #pivotal hd service scripts that shows the service 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'
```

Preparing the Cluster Nodes

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.

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
```

Note: 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.

Note: 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
```

Back to Installation Overview

Pivotal HD Directory Layout

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

Directory Location	Description
*/usr/lib/gphd/**	The default *$GPHD_HOME* folder. This is the default parent folder for Pivotal HD components.
*/etc/gphd/**	The default *$GPHD_CONF* folder. This is the folder for Pivotal HD component configuration files.
/etc/default/	The directory used by service scripts to set up the component environment variables.

/etc/init.d	The location where a components' Linux Service scripts are stored.
*/var/log/gphd/**	The default location of the *$GPHD_LOG directory*. The directory for Pivotal HD component logs.
*/var/run/gphd/**	The location of the any daemon process information for the components.
/usr/bin	The folder for the component's command scripts; only sym-links or wrapper scripts are created here.

Running Sample Programs

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

Testing Hadoop

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 1777. For example if it is set to the default value /yarn/apps, do the following

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

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

Testing HBase

You can test HBase from the HBase master node

```
gpadmin# ./bin/hbase shell
hbase(main):003:0> create 'test', 'cf'
0 row(s) in 1.2200 seconds
hbase(main):003:0> list 'test'
..
1 row(s) in 0.0550 seconds
hbase(main):004:0> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.0560 seconds
hbase(main):005:0> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0370 seconds
hbase(main):006:0> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0450 seconds

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

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

Testing HAWQ

! Use the HAWQ Master node to run HAWQ tests.

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

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

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

gpadmin=# \c gpadmin
You are now connected to database "gpadmin" as user "gpadmin".
```

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

CREATE TABLE

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

gpadmin=#
```

Testing Pig

You can test Pig from the client node

```
# Clean up input/output directories

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

#Create input directory

hadoop fs -mkdir /tmp/test_pig_input

# Copy data from /etc/passwd

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

In the grunt shell, run this simple pig job

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

# Displaying output

hadoop fs -cat /tmp/test_pig_output/part*

Cleaning up input and output'

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

Testing Hive

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;
```

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 directory
hadoop fs -mkdir /tmp/test_uss_input

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

# Add mount-point for input. Do this on the
# uss-client/uss-namenode/uss-catalog host
$ uss admin --add \
  --mount-point-name input_mount_point \
  --mount-point-target \
  hdfs://<NAMENODE_HOST>:8020/tmp/test_uss_input


# Add mount-point for output. Do this on the
# uss-client/uss-namenode/uss-catalog host
$ uss admin --add \
  --mount-point-name output_mount_point \
  --mount-point-target \
  hdfs://NAMENODE_HOST:8020/tmp/test_uss_output

# List mount-points. Do this on the
# uss-client/uss-namenode/uss-catalog host
$ uss 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-0.3.0.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
```

Managing a Cluster

Starting a Cluster

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

```
[gpadmin]# icm_client start -h
Usage: 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
partially complete)
  -w, --reformat        Reformat/Reinitialize the cluster (Namenode
Format &
                        Data Cleanup
  -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.
-f	Forces the cluster to start even if the installation is incomplete.

-w	Reformats the namenode and data directories before starting the cluster. Use this option if you want to wipe out the existing state of the cluster data and start from scratch.

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.

! 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.

> **!** Please refer to the "Verifying the Cluster Nodes for Pivotal HD" section to make sure the cluster services are up and running.

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
```

#PivotalHDManagerUserGuide-ArtemisRelease-VerifyClusterInstallation

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 which 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
```

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 *htt p-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: 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
```

To reconfigure an existing cluster:

1. Stop the cluster:
   icm_client stop -l CLUSTERNAME
2. Fetch the configurations for the cluster in a local directory:
   icm_client fetch-configuration -l CLUSTERNAME -o LOCALDIR
3. Edit the configuration files in the cluster configuration directory (LOCALDIR).
4. Reconfigure the cluster:
   icm_client reconfigure -l CLUSTERNAME -c LOCALDIR

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

1. Fetch the new templates that come with the upgraded software by running *icm_client*

fetch-template.
2. Retrieve the existing configuration from database using *icm_client fetch-configuration*.
3. **Synchronize the new configurations (hdfs/hadoop-env) from the template directory to the existing cluster configuration directory.**
4. Upgrade or reconfigure service by specifying the cluster configuration directory with updated contents.

Add / Remove Services

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

- Edit the clusterConfig.xml file to add / 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

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.

Retrieving Configuration about a Deployed Cluster

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

```
icm_client fetch-configuration -h
Usage: icm_client fetch-configuration [options]

Options:
  -h, --help            show this help message and exit
  -o OUTDIR, --outdir=OUTDIR
                        Directory path to store the cluster configuration
                        template files
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        Name of the deployed cluster whose configurations
need
                        to be fetched
```

Sample Usage

icm_client fetch-configuration -l CLUSTERNAME -o LOCALDIR

Listing Clusters

Run the *list* command to see a list of all the installed clusters

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

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

Sample Usage:

icm_client list

Expanding a Cluster

! Please 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.

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

Options:
  -h, --help      show this help message and exit
  -v, --verbose   increase output verbosity  -v, --verbose  increase
output verbosity
  -f, --file      file containing the new slave hosts to be added

  -j, --java_home JAVA_HOME path to verify on cluster nodes
```

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 please start hbase-regionservers as well.

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

Shrinking a Cluster

! Please 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.

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

Options:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
  -f, --file     file containing the slave hosts to be removed
```

Sample Usage:

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

Decommissioning Nodes

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

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

- On the NameNode host machine
  - Edit the */etc/gphd/hadoop/conf/dfs.exclude* file and add the datanode hostnames to be removed (separated by newline character). Make sure you use 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:50070and 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 -h hostfile
[gpadmin] icm_client stop -l CLUSTERNAME -r yarn-nodemanager -h 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.

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.**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.

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*.

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* f or instructions

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
```

Stopping HAWQ

Run the *stop* command to stop the hawq master, segments hosts, and the postgres database on the HAWQ master.

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

Modifying HAWQ User Configuration

If you are using 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
   ```

4. 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;"
```

Upgrading a Cluster

Pivotal HD only supports upgrading cluster services that were originally installed using PCC's CLI. Services that were manually installed on an existing cluster will not be available post upgrade.

Steps to upgrade manually installed services

We recommend you backup your data before proceeding with *any* upgrades.

- Backup the service's configuration files
- Perform the cluster upgrade (Pivotal HD or ADS) as shown below
- Reinstall the service
- Restore the service's configuration files across the cluster

For information about manually installing services and for the locations of services' configuration files, see the *Pivotal HD Enterprise Stack and Tool Reference Guide.*

Note:

- If you were a PCC 2.0.0 user, Hive and GPXF were not automatically installed. If you manually installed those services on a cluster, you must follow the steps above.
- If you are upgrading to 1.0.2 or higher, the following configurations have to be ensured for hive, hbase and pxf to work. This can be done by running "icm_client fetch_template" command to get all the latest env scripts and xml files and then doing a manual sync with the output of "icm_client fetch-configuration" that is used for running upgrade.

```
file to check if output of fetch-configuration is in
fetchedClusterConfigDir: fetchedClusterConfigDir/2.0/hdfs/hadoop-env.sh

# 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 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_HOME/hive/conf
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:\
```

```
$HIVELIB_HOME/hive-service-0.11.0-gphd-2.0.3.0.jar:\
$HIVELIB_HOME/libthrift-0.9.0.jar:\
$HIVELIB_HOME/hive-metastore-0.11.0-gphd-2.0.3.0.jar:\
$HIVELIB_HOME/libfb303-0.9.0.jar:\
$HIVELIB_HOME/hive-common-0.11.0-gphd-2.0.3.0.jar:\
$HIVELIB_HOME/hive-exec-0.11.0-gphd-2.0.3.0.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:\
```

Upgrade Syntax

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

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

Options:
  -h, --help            show this help message and exit
  -l CLUSTERNAME, --clustername=CLUSTERNAME
                        the name of the cluster on which the operation is
                        performed
  -s STACK, --stackname=STACK
                        stack to upgrade (Pivotal HD or ADS)
  -c CONFDIR, --confdir=CONFDIR
                        (Required only for Pivotal HD upgrade) Directory
path where
                        cluster configuration is stored
  -o OLDDIR, --old=OLDDIR
                        (Required for only for Pivotal ADS/HAWQ upgrade)
Old PADS
                        Directory
  -n NEWDIR, --new=NEWDIR
                        (Required for only for Pivotal ADS/HAWQ upgrade)
New PADS
                        Directory
```

Upgrade PADS (HAWQ)

Prior to upgrading HAWQ do the following

- Download and Import the required new Pivotal ADS Stack.
- Stop the HAWQ Master.

***Sample Usage** :*

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

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

Upgrade Pivotal HD

Prior to upgrading the Pivotal HD stack, make sure you do the following

- Download and import the new Pivotal HD Stack on the Admin Node
- Stop the Cluster
- If the cluster is configured with HAWQ, make sure you complete upgrading Pivotal ADS (see above), before proceeding with Pivotal HD upgrade.

Sample Usage:
icm_client upgrade -l <CLUSTERNAME> -s phd -c ClusterConfigDir/

Once the Upgrade is done, you need to update the nmon config file manually on the Admin node for the System Monitoring component to work with the newly upgraded stacks.

```
service nmon stop
Change the port information in /etc/gphd/gphdmgr/conf/nmon-site.xml from
5432 to 10432
service nmon start
```

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.

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

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, follow these steps:

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

- Managing locally
- Managing from the Admin Node

Managing Locally

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

```
node100:gpadmin# ssh gpadmin@node100
gpadmin# sudo service hadoop-hdfs-namenode restart
```

Managing Remotely

You can manage the service role remotely across one of the target hosts. For example, to restart datanode:

```
node100.gpadmin# massh node100 verbose 'sudo service hadoop-hdfs-datanode
restart'
```

To restart all the datanodes remotely:

Create a newline separated file named 'hostfile' containing all the datanodes to *start*, *stop*, *r estart*, or *check* status.

```
gpadmin# massh hostfile verbose 'sudo service hadoop-hdfs-datanode
restart'
```

The following table shows the service commands to *start*, *stop*, *restart*, or *check* status for each service role,.

Role Name	Service Command
Namenode	`sudo service hadoop-hdfs-namenode {starts|stop|status|restart}`
Secondary NameNode	`sudo service hadoop-hdfs-secondarynamenode {starts|stop|status|restart}`
Datanode	`sudo service hadoop-hdfs-datanode {starts|stop|status|restart}`
Resource Manager	`sudo service hadoop-yarn-resourcemanager {starts|stop|status|restart}`

Node Manager	`sudo service hadoop-yarn-nodemanager {starts|stop|status|restart}`
History Server	`sudo service hadoop-mapreduce-historyserver {starts|stop|status|restart}`
Zookeeper Server	`sudo service zookeeper-server {starts|stop|status|restart}`
HBase Master	`sudo service hbase-master {starts|stop|status|restart}`
HBase Region Server	`sudo service hbase-regionserver {starts|stop|status|restart}`
HAWQ Master	`sudo /etc/init.d/hawq {starts|stop|status|restart}`
USS Namenode	`sudo /etc/init.d/uss-namenode {start|stop|status|restart}`

Pivotal HD Services Reference

Overriding Directory Permissions

The following table shows the list of directories that the 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
YARN	*mapreduce.cluster.local.dir*	All yarn nodes	*mapred:hadoop*	755
	mapreduce.cluster.temp.dir	All yarn nodes	*mapred:hadoop*	755
	yarn.nodemanager.local-dirs	Node Managers	*yarn:yarn*	755
	yarn.nodemanager.log-dirs	Node Managers	*yarn:yarn*	755
ZooKeeper	*dataDir (/var/lib/zookeeper)*	Zookeeper Servers	*zookeeper:zookeeper*	775
	dataDir/myid	Zookeeper Servers	*gpadmin*	644
HAWQ	*MASTER_DIRECTORY*	HAWQ Master & Standby	*gpadmin:hadoop*	755
	DATA_DIRECTORY	HAWQ Segments	*gpadmin:hadoop*	755

On HDFS

Service	Directory	Owner	Permissions
HDFS	*hadoop.tmp.dir*	*hdfs:hadoop*	777
	/tmp	*hdfs:hadoop*	777
	mapreduce.jobtracker.system.dir	*mapred:hadoop*	700
	yarn.app.mapreduce.am.staging-dir (/user)	*mapred:hadoop*	777
	mapreduce.jobhistory.intermediate-done-dir (/user/history/done)	*mapred:hadoop*	777
	mapreduce.jobhistory.done-dir (/user/history/done)	*mapred:hadoop*	777
	yarn.nodemanager.remote-app-log-dir	*mapred:hadoop*	755 (should be 777)
HBase	*hbase directory (/apps/hbase/data)*	*hdfs:hadoop*	775
HAWQ	*hawq directory (/hawq_data)*	*hdfs:hadoop*	755

Pivotal HD Users and Groups

Service	Users	Group	Login
PHD	gpadmin	gpadmin	Yes
HDFS	hdfs	hadoop	Yes
MapReduce	mapred	hadoop	Yes
	hadoop	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
PXF			
DataLoader	dladmin	dladmin	Yes
	dataloader	root	No

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

Pivotal HD Ports

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

Service	Port
ssh	22
NameNode	8020 (Apache 9000)
NameNode Web UI	50070, 50470 (https)
Secondary NameNode	50090
DataNode Communication	50010
DataNode IPC	50020
DataNode HTTP Address	50075
ResourceManager Web UI	8042,8088

ResourceManager	8030,8031,8032,8033
MapReduce Shuffle Port	7070
Job History Server	10020
Job History Web UI	19888
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

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 softwares. 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.

Frequently Asked Questions

Can I deploy multiple clusters from the same admin?

Yes, you can deploy any number of Pivotal HD clusters from the same admin. You must deploy them in succession, not simultaneously.

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

No, you cannot change the topology.

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.

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

What group and users are created by Pivotal HD?

Please refer to the Troubleshooting section for details about the users and directories created by PCC.

What is the allowed time difference amongst the cluster nodes 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.

Does PCC support simultaneous deployment of multiple clusters?

No. Concurrent deployment is not allowed. Please wait till the first deployment is complete before starting another.

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

No, only FQDN format is currently supported.

Can a node be shared between different clusters?

No, nodes cannot be shared between clusters

I installed puppet-2.7.20 from the Puppet Labs repository but Pivotal HD does not work?

Pivotal HD requires the version of puppet shipped with the product and not the downloadable version from the Puppet Labs repository. Uninstall Puppet and install the one shipped with the product using the *icm_client preparehosts* command.

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

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

Will I lose my data if i uninstall the cluster?

Uninstalling the cluster will not wipe out any data. But a subsequent installation would wipe out the configured mount points upon confirmation. Make sure you back out the data.

Will i lose my data if I upgrade the PHD/ADS stack through the stack import utility?

Upgrading any stack using the import utility will not affect your cluster/data as long as the upgrade is compatible with the existing data layout.

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

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

How do I change the port used by Pivotal HD?

1. Log onto the machine as root.
2. Stop 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
```

Then

```
\#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
```

Troubleshooting

Debugging Errors

Pivotal Command Center has many different log files. Finding the exact log may initially be challenging at the beginning.

Here is a quick guide on how to identify the issues:

Pivotal HD Installation

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

Cluster Deployment

If you see a 500 Internal Server Error, check the following logs for details:
/var/log/gphd/gphdmgr/gphdmgr-webservices.log

If you see Puppet cert generation error, 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

Cluster Nodes Installation

If there are no errors on the admin side, but the installation failed on the cluster nodes, check the latest log file:
/tmp/GPHDNodeInstaller_XXX.log

Search for the first occurrence of the word "merr" that will point to the most probable issue.

Services Start

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

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

Puppet SSL Errors

For errors like:
"Unable to generate certificates"
"SSLv3 authentication issues on the client"

As root, do the following:

Ensure the hostname on all machines is FQDN. 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.

Make sure there are no certificates listed by running:

```
puppetca list --all
```

You can run *puppetca clean --all* to clean any certificates

Restart the puppet master:

```
service puppetmaster start
```

Verify there is just one certificate:

```
puppetca list --all
```

Stop the puppet master and start nmon:

```
service puppetmaster stop

service commander start
```

Now retry your deployment.

Upgrade /Reconfigure Errors

Following an upgrade of Command Center, unable to Start/Stop cluster with invalid hostnames.

This is because there is now a check for invalid characters in cluster names.
Workaround: First reconfigure the cluster to a different name:

```
icm_client reconfigure -l <old_cluster_name> -c <config directory with
new clustername>
```

Then try starting/stopping the cluster:

```
icm_client start -l <cluster_name>
icm_client stop -l <cluster_name>
```

Other Upgrade/Reconfigure Errors

- After upgrading PHD stack from 1.0.2 to 1.0.3 release, hbase master fails to start if hbase-master is not co-located with either namenode or datanode.
  Workaround: On hbase-master node, run "yum upgrade hadoop-hdfs". Go to /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 please refer to the Note in "Upgrading a Cluster" section and perform the manual steps listed.

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

Unable to access the Namenode Status Web page

If the host returns a short hostname instead of FQDN for hostname(), it is possible that the namenode status link cannot be accessed from external networks.

The solution is to either ensure that the hostname() returns FQDN on the namenode host (or) Change the "dfs.http.address" value to 0.0.0.0 in the hdfs-site.xml and restart namenode<property>
<name>dfs.http.address</name>
<value>0.0.0.0:50070</value>
</property>

Installation Fails due to Directory Permissions

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.

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.

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.

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

Puppet certificate failure

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

Package Bundle Not Found

If you sudo in to the system as root, please ensure that you sudo with the environment. i.e, "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

Cluster Deployment Fails due to Missing Packages

The above error can be identified by following the instructions on "Cluster Nodes Installation Errors" section above.

Install *nc* and *postgres-devel* packages on all the cluster nodes or point them to a repo that contains the rpms.

Working with Proxy Servers

It 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 work arounds:

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/}
```

Capital letters in hostname

PCC fails to deploy if the hostnames contain uppercase letters. For example: "Node0781.domain.com".
Rename the hostname with only lowercase letters before proceeding with the deployment.

Resolving postgres port conflict issue

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

Resolving HTTP port conflict

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

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.

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 2.0 Installation and User Guide* for instructions to disable SELinux)

HAWQ initialization failing

Make sure your cluster is up and running with the hadoop services prior to 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.

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

1. 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.

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).