Pivotal

PRODUCT DOCUMENTATION

Pivotal™ Greenplum Database®

Version 4.3

Installation Guide

Rev: A31

© 2018 Pivotal Software, Inc.

Notice

Copyright

Privacy Policy | *Terms of Use*

Copyright © 2018 Pivotal Software, Inc. All rights reserved.

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

Use, copying, and distribution of any Pivotal software described in this publication requires an applicable software license.

All trademarks used herein are the property of Pivotal or their respective owners.

Revised November 2018 (4.3.30.3)

Contents

Chapter 1

Preface

This guide describes the tasks you must complete to install and start your Greenplum Database system.

About This Guide

This guide provides information and instructions for installing and initializing a Greenplum Database system. This guide is intended for system administrators responsible for building a Greenplum Database system.

This guide assumes knowledge of Linux/Unix system administration, database management systems, database administration, and structured query language (SQL).

This guide contains the following chapters and appendices:

- *Introduction to Greenplum* — Information about the Greenplum system architecture and components.
- *Estimating Storage Capacity* — Guidelines for sizing a Greenplum Database system.
- *Configuring Your Systems and Installing Greenplum* — Instructions for installing and configuring the Greenplum software on all hosts in your Greenplum Database array.
- *Validating Your Systems* — Validation utilities and tests you can perform to ensure your Greenplum Database system will operate properly.
- *Configuring Localization Settings* — Localization features of Greenplum Database. Locale settings must be configured prior to initializing your Greenplum Database system.
- *Initializing a Greenplum Database System* — Instructions for initializing a Greenplum Database system. Each database instance (the master and all segments) must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS.
- *Installation Management Utilities* — Reference information about the command-line management utilities you use to install and initialize a Greenplum Database system.
- *Greenplum Environment Variables* — Reference information about Greenplum environment variables you can set in your system user's profile file.

About the Greenplum Database Documentation Set

The Greenplum Database 4.3 server documentation set consists of the following guides.

Table 1: Greenplum Database server documentation set

Guide Name	Description
Greenplum Database Administrator Guide	Information for administering the Greenplum Database system and managing databases. It covers topics such as Greenplum Database architecture and concepts and everyday system administration tasks such as configuring the server, monitoring system activity, enabling high-availability, backing up and restoring databases, and expanding the system. Database administration topics include configuring access control, creating databases and database objects, loading data into databases, writing queries, managing workloads, and monitoring and troubleshooting performance.
Greenplum Database Reference Guide	Reference information for Greenplum Database systems: SQL commands, system catalogs, environment variables, character set support, datatypes, the Greenplum MapReduce specification, postGIS extension, server parameters, the gp_toolkit administrative schema, and SQL 2008 support.
Greenplum Database Utility Guide	Reference information for command-line utilities, client programs, and Oracle compatibility functions.
Greenplum Database Installation Guide	Information and instructions for installing and initializing a Greenplum Database system.

Document Conventions

Greenplumdocumentation adheres to the following conventions to help you identify certain types of information.

- *Command Syntax Conventions*

Command Syntax Conventions

Table 2: Command Syntax Conventions

Text Convention	Usage	Examples
`{ }`	Within command syntax, curly braces group related command options. Do not type the curly braces.	`FROM { ' filename ' | STDIN }`
`[ ]`	Within command syntax, square brackets denote optional arguments. Do not type the brackets.	`TRUNCATE [ TABLE ] name`
`...`	Within command syntax, an ellipsis denotes repetition of a command, variable, or option. Do not type the ellipsis.	`DROP TABLE name [, ... ]`
`|`	Within command syntax, the pipe symbol denotes an "OR" relationship. Do not type the pipe symbol.	`VACUUM [ FULL | FREEZE ]`

`$` system_command
`#` root_system_command
`=>` gpdb_command
`=#` su_gpdb_command
	Denotes a command prompt - do not type the prompt symbol. `$` and `#` denote terminal command prompts. `=>` and `=#` denote Greenplum Database interactive program command prompts (`psql` or `gpssh`, for example).	`$ createdb mydatabase`
		`# chown gpadmin -R /datadir`
		`=> SELECT * FROM mytable;`
		`=# SELECT * FROM pg_database;`

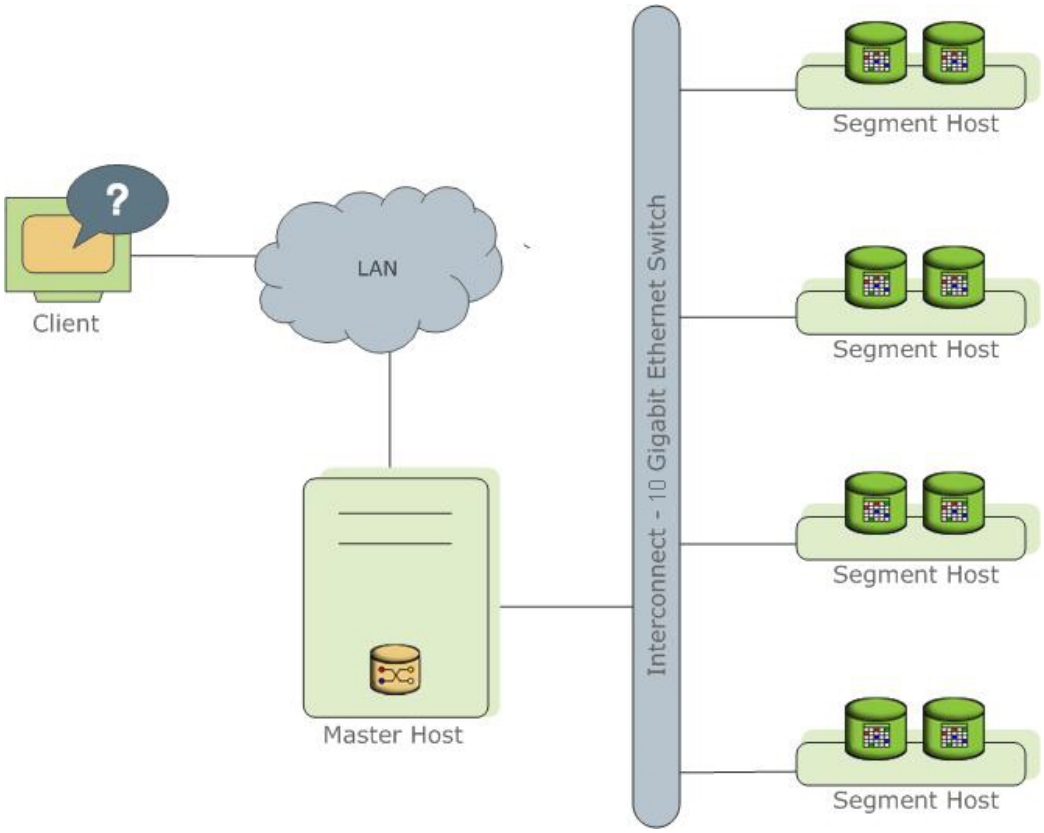
Getting Support

For technical support, documentation, release notes, software updates, or for information about Pivotal products, licensing, and services, go to *www.pivotal.io*.

Chapter 2

Introduction to Greenplum

Greenplum Database stores and processes large amounts of data by distributing the load across several servers or *hosts*. A logical database in Greenplum is an *array* of individual PostgreSQL databases working together to present a single database image. The *master* is the entry point to the Greenplum Database system. It is the database instance to which users connect and submit SQL statements. The master coordinates the workload across the other database instances in the system, called *segments*, which handle data processing and storage. The segments communicate with each other and the master over the *interconnect*, the networking layer of Greenplum Database.



Greenplum Database is a software-only solution; the hardware and database software are not coupled. Greenplum Database runs on a variety of commodity server platforms from Greenplum-certified hardware vendors. Performance depends on the hardware on which it is installed. Because the database is distributed across multiple machines in a Greenplum Database system, proper selection and configuration of hardware is vital to achieving the best possible performance.

This chapter describes the major components of a Greenplum Database system and the hardware considerations and concepts associated with each component: *The Greenplum Master*, *The Segments* and *The Interconnect*. Additionally, a system may have optional *ETL Hosts for Data Loading* and the *Greenplum Performance Monitoring* for monitoring query workload and performance.

The Greenplum Master

The *master* is the entry point to the Greenplum Database system. It is the database server process that accepts client connections and processes the SQL commands that system users issue. Users connect to Greenplum Database through the master using a PostgreSQL-compatible client program such as `psql` or ODBC.

The master maintains the *system catalog* (a set of system tables that contain metadata about the Greenplum Database system itself), however the master does not contain any user data. Data resides only on the *segments*. The master authenticates client connections, processes incoming SQL commands, distributes the work load between segments, coordinates the results returned by each segment, and presents the final results to the client program.

Because the master does not contain any user data, it has very little disk load. The master needs a fast, dedicated CPU for data loading, connection handling, and query planning because extra space is often necessary for landing load files and backup files, especially in production environments. Customers may decide to also run ETL and reporting tools on the master, which requires more disk space and processing power.

Master Redundancy

You may optionally deploy a *backup* or *mirror* of the master instance. A backup master host serves as a *warm standby* if the primary master host becomes nonoperational. You can deploy the standby master on a designated redundant master host or on one of the segment hosts.

The standby master is kept up to date by a transaction log replication process, which runs on the standby master host and synchronizes the data between the primary and standby master hosts. If the primary master fails, the log replication process shuts down, and an administrator can activate the standby master in its place. When an the standby master is active, the replicated logs are used to reconstruct the state of the master host at the time of the last successfully committed transaction.

Since the master does not contain any user data, only the system catalog tables need to be synchronized between the primary and backup copies. When these tables are updated, changes automatically copy over to the standby master so it is always synchronized with the primary.

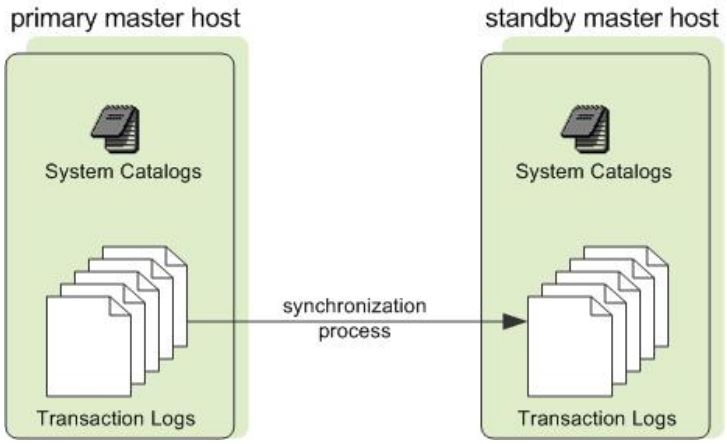


Figure 1: Master Mirroring in Greenplum Database

The Segments

In Greenplum Database, the *segments* are where data is stored and where most query processing occurs. User-defined tables and their indexes are distributed across the available segments in the Greenplum Database system; each segment contains a distinct portion of the data. Segment instances are the database server processes that serve segments. Users do not interact directly with the segments in a Greenplum Database system, but do so through the master.

In the reference Greenplum Database hardware configurations, the number of segment instances per segment host is determined by the number of effective CPUs or CPU core. For example, if your segment hosts have two dual-core processors, you may have two or four primary segments per host. If your segment hosts have three quad-core processors, you may have three, six or twelve segments per host. Performance testing will help decide the best number of segments for a chosen hardware platform.

Segment Redundancy

When you deploy your Greenplum Database system, you have the option to configure *mirror* segments. Mirror segments allow database queries to fail over to a backup segment if the primary segment becomes unavailable. Mirroring is a requirement for Pivotal-supported production Greenplum Database systems.

A mirror segment must always reside on a different host than its primary segment. Mirror segments can be arranged across the hosts in the system in one of two standard configurations, or in a custom configuration you design. The default configuration, called *group* mirroring, places the mirror segments for all primary segments on a host on one other host. Another option, called *spread* mirroring, spreads mirrors for each host's primary segments over the remaining hosts. Spread mirroring requires that there be more hosts in the system than there are primary segments on the host. On hosts with multiple network interfaces, the primary and mirror segments are distributed equally among the interfaces. *Figure 2: Data Mirroring in Greenplum Database* shows how table data is distributed across the segments when the default group mirroring option is configured.

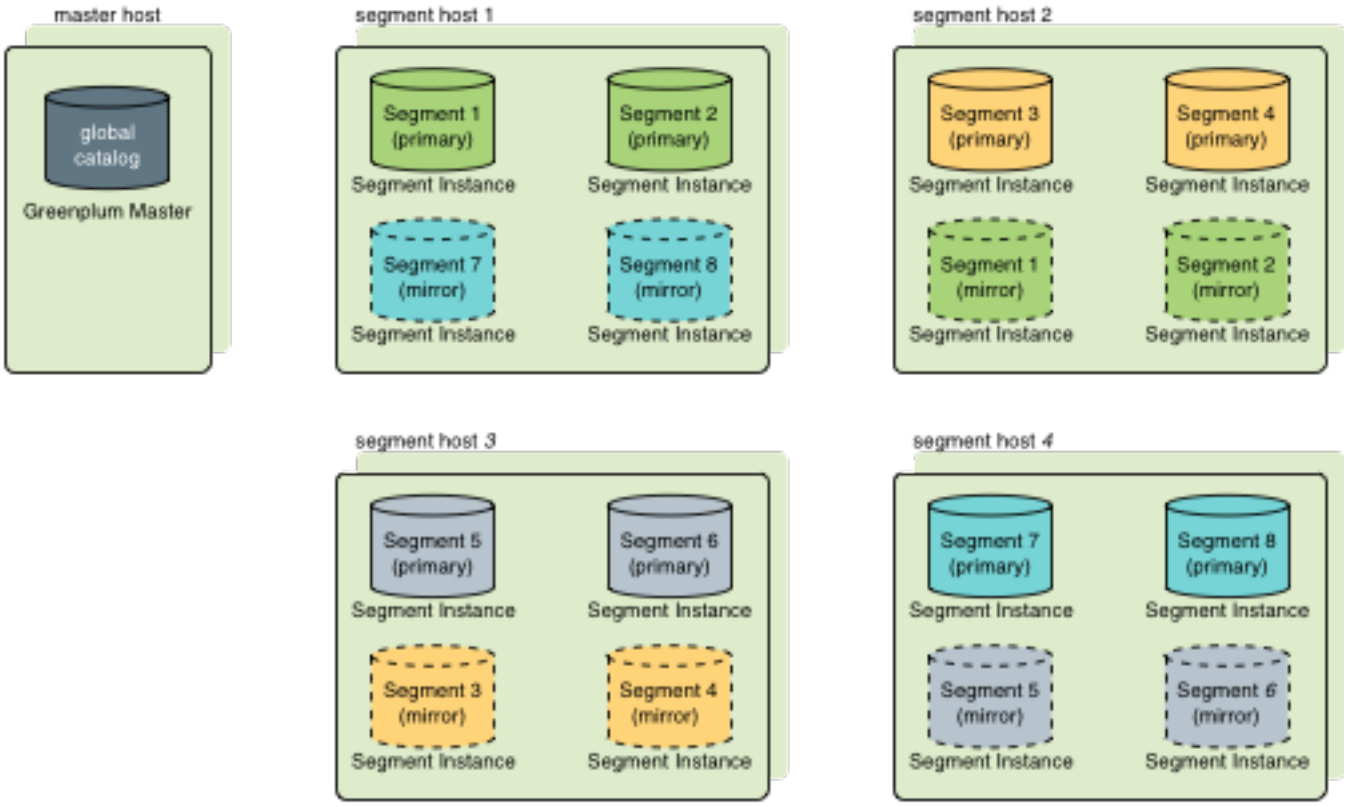


Figure 2: Data Mirroring in Greenplum Database

Segment Failover and Recovery

When mirroring is enabled in a Greenplum Database system, the system automatically fails over to the mirror copy if a primary copy becomes unavailable. A Greenplum Database system can remain operational if a segment instance or host goes down only if all portions of data are available on the remaining active segments.

If the master cannot connect to a segment instance, it marks that segment instance as *invalid* in the Greenplum Database system catalog. The segment instance remains invalid and out of operation until an administrator brings that segment back online. An administrator can recover a failed segment while the system is up and running. The recovery process copies over only the changes that were missed while the segment was nonoperational.

If you do not have mirroring enabled and a segment becomes invalid, the system automatically shuts down. An administrator must recover all failed segments before operations can continue.

Example Segment Host Hardware Stack

Regardless of the hardware platform you choose, a production Greenplum Database processing node (a segment host) is typically configured as described in this section.

The segment hosts do the majority of database processing, so the segment host servers are configured in order to achieve the best performance possible from your Greenplum Database system. Greenplum Database's performance will be as fast as the slowest segment server in the array. Therefore, it is important to ensure that the underlying hardware and operating systems that are running Greenplum Database are all running at their optimal performance level. It is also advised that all segment hosts in a Greenplum Database array have identical hardware resources and configurations.

Segment hosts should also be dedicated to Greenplum Database operations only. To get the best query performance, you do not want Greenplum Database competing with other applications for machine or network resources.

The following diagram shows an example Greenplum Database segment host hardware stack. The number of effective CPUs on a host is the basis for determining how many primary Greenplum Database segment instances to deploy per segment host. This example shows a host with two effective CPUs (one dual-core CPU). Note that there is one primary segment instance (or primary/mirror pair if using mirroring) per CPU core.

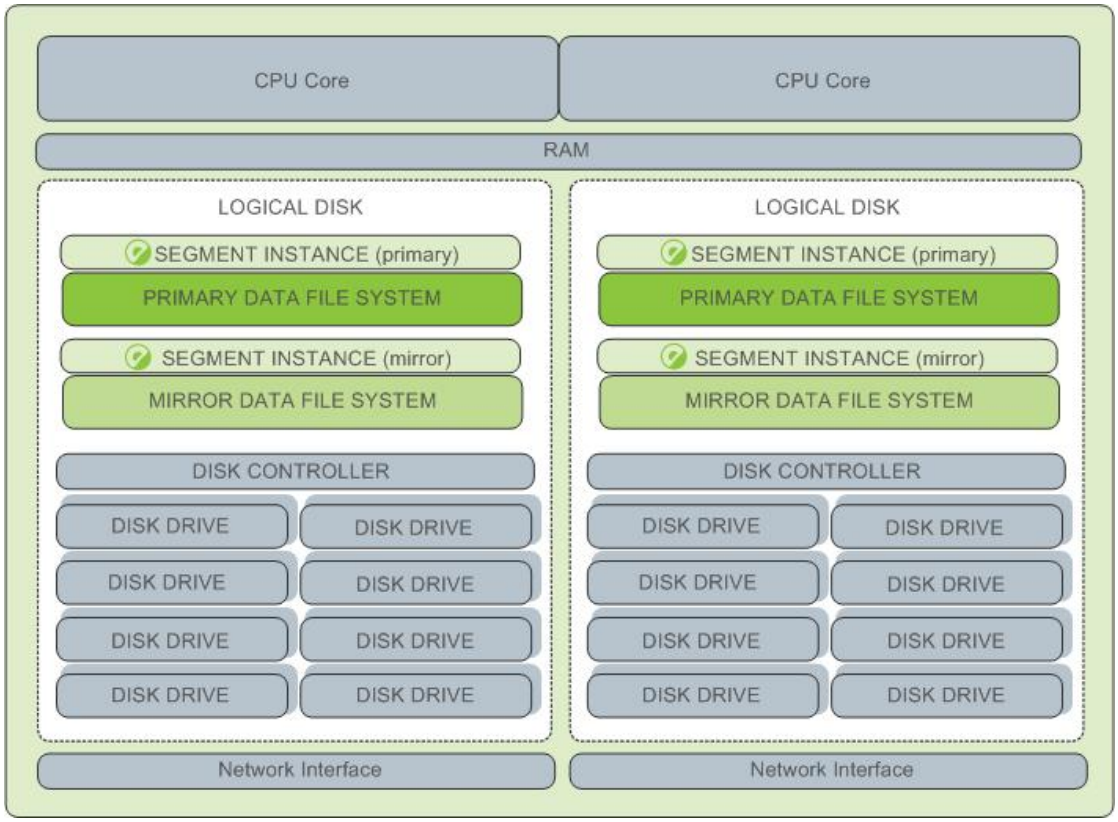


Figure 3: Example Greenplum Database Segment Host Configuration

Example Segment Disk Layout

Each CPU is typically mapped to a logical disk. A logical disk consists of one primary file system (and optionally a mirror file system) accessing a pool of physical disks through an I/O channel or disk controller. The logical disk and file system are provided by the operating system. Most operating systems provide the ability for a logical disk drive to use groups of physical disks arranged in RAID arrays.

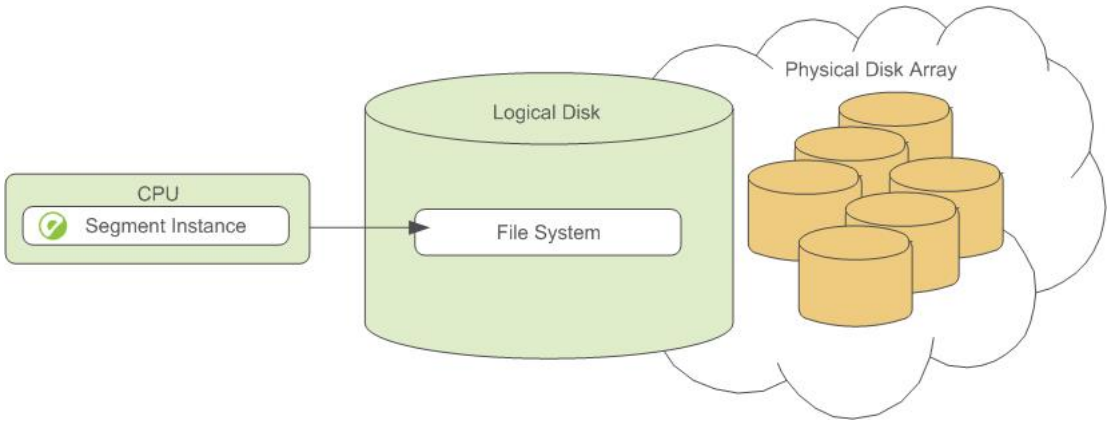


Figure 4: Logical Disk Layout in Greenplum Database

Depending on the hardware platform you choose, different RAID configurations offer different performance and capacity levels. Greenplum supports and certifies a number of reference hardware platforms and operating systems. Check with your sales account representative for the recommended configuration on your chosen platform.

The Interconnect

The *interconnect* is the networking layer of Greenplum Database. When a user connects to a database and issues a query, processes are created on each of the segments to handle the work of that query. The *interconnect* refers to the inter-process communication between the segments, as well as the network infrastructure on which this communication relies. The interconnect uses a standard 10 Gigabit Ethernet switching fabric.

By default, Greeplum Database interconnect uses UDP (User Datagram Protocol) with flow control for interconnect traffic to send messages over the network. The Greenplum software does the additional packet verification and checking not performed by UDP, so the reliability is equivalent to TCP (Transmission Control Protocol), and the performance and scalability exceeds that of TCP. For information about the types of interconnect supported by Greeplum Database, see server configuration parameter `gp_interconnect_type` in the *Greenplum Database Reference Guide*.

Interconnect Redundancy

A highly available interconnect can be achieved by deploying dual 10 Gigabit Ethernet switches on your network, and redundant 10 Gigabit connections to the Greenplum Database master and segment host servers.

Network Interface Configuration

A segment host typically has multiple network interfaces designated to Greenplum interconnect traffic. The master host typically has additional external network interfaces in addition to the interfaces used for interconnect traffic.

Depending on the number of interfaces available, you will want to distribute interconnect network traffic across the number of available interfaces. This is done by assigning segment instances to a particular network interface and ensuring that the primary segments are evenly balanced over the number of available interfaces.

This is done by creating separate host address names for each network interface. For example, if a host has four network interfaces, then it would have four corresponding host addresses, each of which maps to one or more primary segment instances. The `/etc/hosts` file should be configured to contain not only the host name of each machine, but also all interface host addresses for all of the Greenplum Database hosts (master, standby master, segments, and ETL hosts).

With this configuration, the operating system automatically selects the best path to the destination. Greenplum Database automatically balances the network destinations to maximize parallelism.

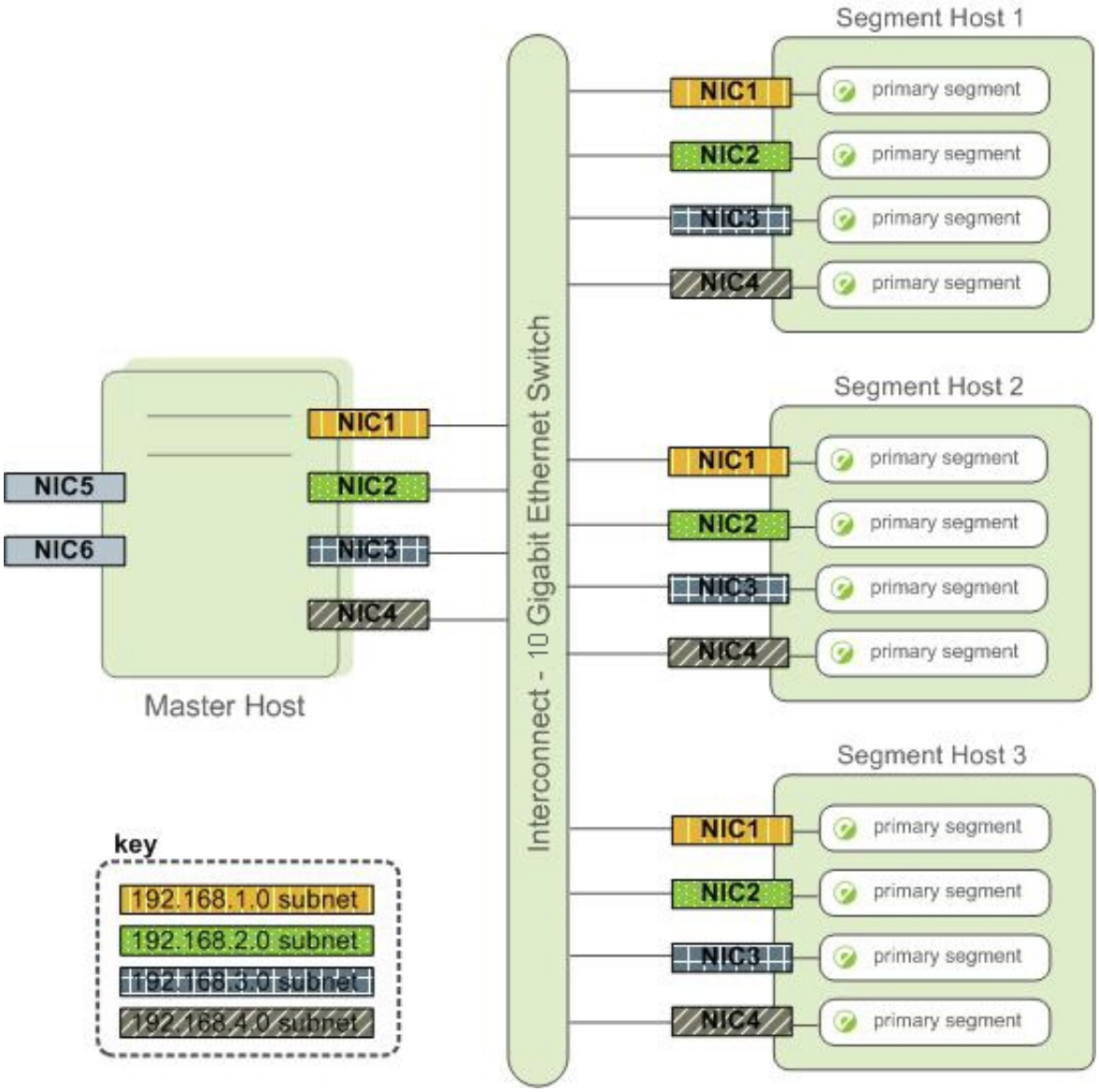


Figure 5: Example Network Interface Architecture

Switch Configuration

When using multiple 10 Gigabit Ethernet switches within your Greenplum Database array, evenly divide the number of subnets between each switch. In this example configuration, if we had two switches, NICs 1 and 2 on each host would use switch 1 and NICs 3 and 4 on each host would use switch 2. For the master host, the host name bound to NIC 1 (and therefore using switch 1) is the effective master host name for the array. Therefore, if deploying a warm standby master for redundancy purposes, the standby master should map to a NIC that uses a different switch than the primary master.

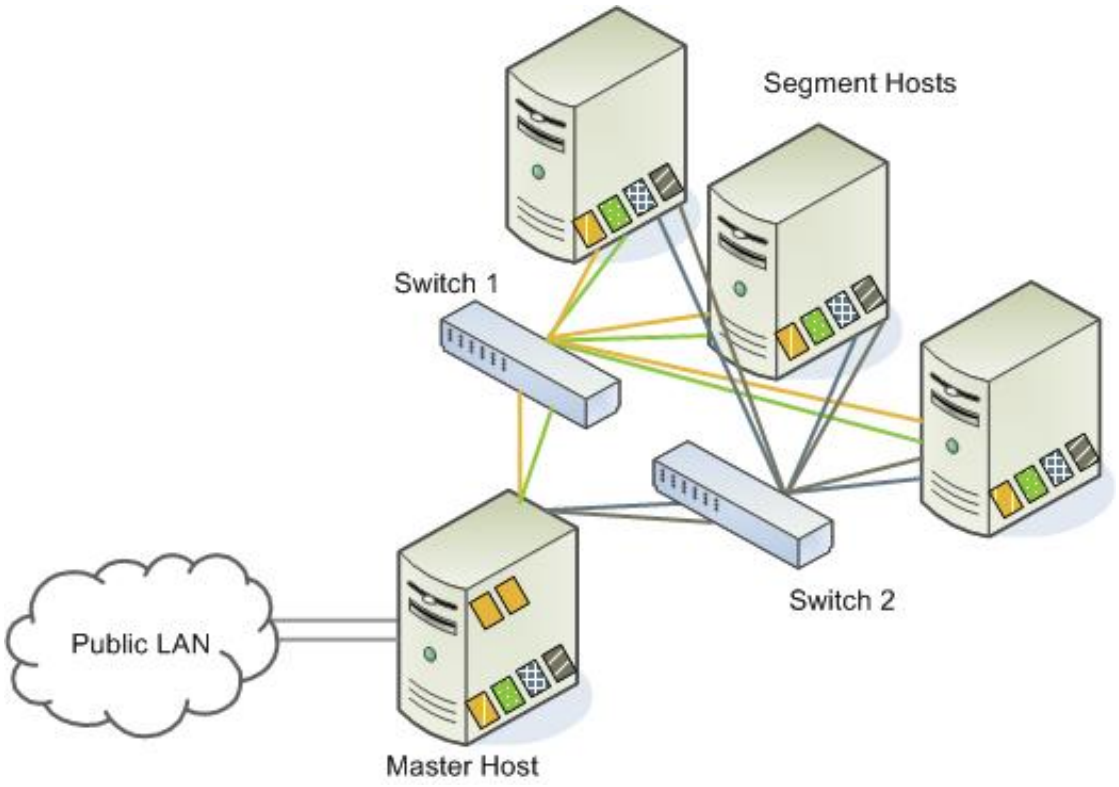


Figure 6: Example Switch Configuration

ETL Hosts for Data Loading

Greenplum supports fast, parallel data loading with its external tables feature. By using external tables in conjunction with Greenplum Database's parallel file server (`gpfdist`), administrators can achieve maximum parallelism and load bandwidth from their Greenplum Database system. Many production systems deploy designated ETL servers for data loading purposes. These machines run the Greenplum parallel file server (`gpfdist`), but not Greenplum Database instances.

One advantage of using the `gpfdist` file server program is that it ensures that all of the segments in your Greenplum Database system are fully utilized when reading from external table data files.

The `gpfdist` program can serve data to the segment instances at an average rate of about 350 MB/s for delimited text formatted files and 200 MB/s for CSV formatted files. Therefore, you should consider the following options when running `gpfdist` in order to maximize the network bandwidth of your ETL systems:

- If your ETL machine is configured with multiple network interface cards (NICs) as described in *Network Interface Configuration*, run one instance of `gpfdist` on your ETL host and then define your external table definition so that the host name of each NIC is declared in the `LOCATION` clause (see `CREATE EXTERNAL TABLE` in the *Greenplum Database Reference Guide*). This allows network traffic between your Greenplum segment hosts and your ETL host to use all NICs simultaneously.

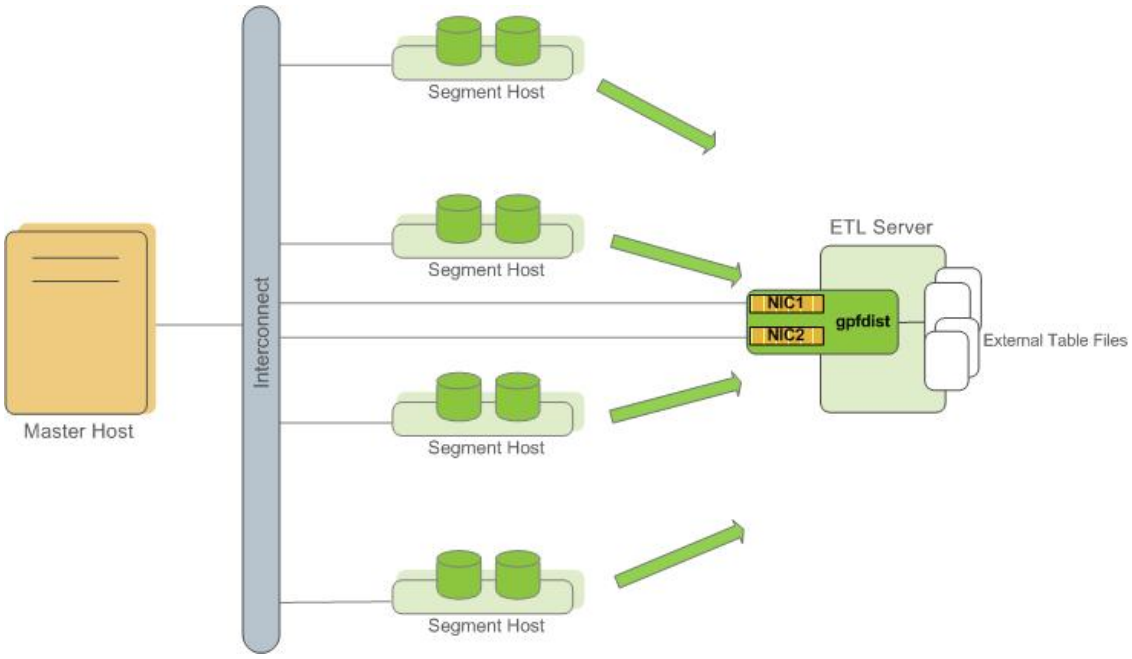


Figure 7: External Table Using Single gpfdist Instance with Multiple NICs

- Run multiple `gpfdist` instances on your ETL host and divide your external data files equally between each instance. For example, if you have an ETL system with two network interface cards (NICs), then you could run two `gpfdist` instances on that machine to maximize your load performance. You would then divide the external table data files evenly between the two `gpfdist` programs.

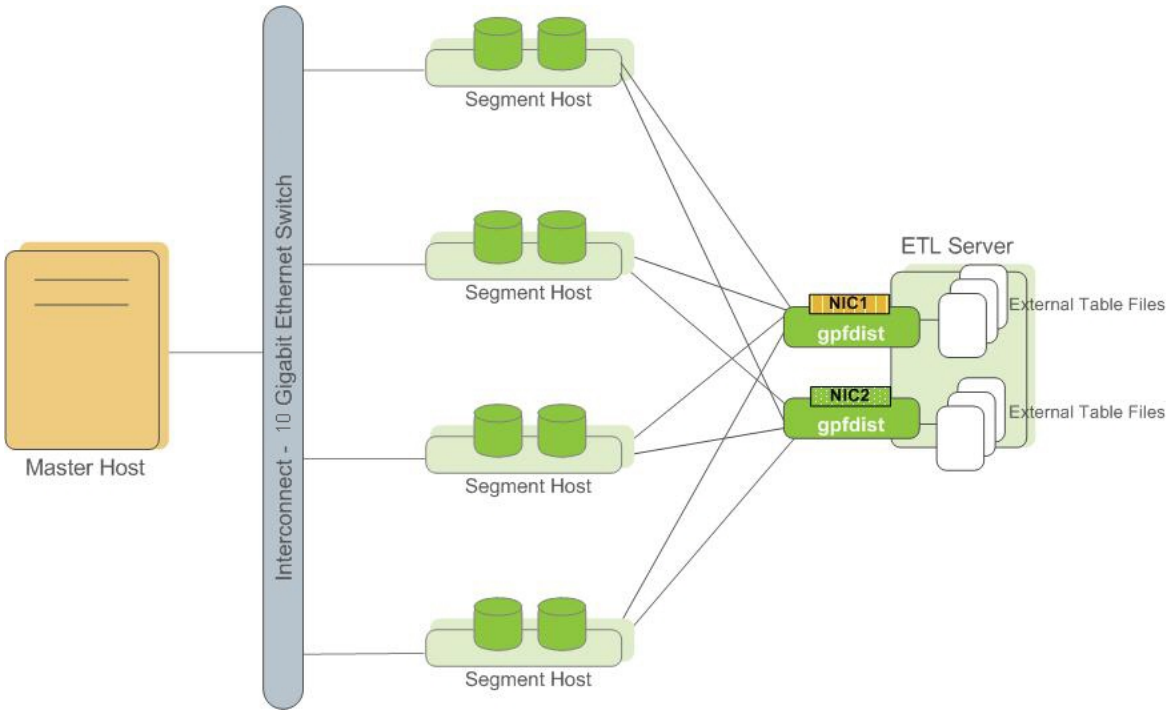


Figure 8: External Tables Using Multiple gpfdist Instances with Multiple NICs

Greenplum Performance Monitoring

Greenplum Database includes a dedicated system monitoring and management database, named gpperfmon, that administrators can install and enable. When this database is enabled, data collection agents on each segment host collect query status and system metrics. At regular intervals (typically every 15 seconds), an agent on the Greenplum master requests the data from the segment agents and updates the gpperfmon database. Users can query the gpperfmon database to see the stored query and system metrics. For more information see the "gpperfmon Database Reference" in the *Greenplum Database Reference Guide*.

Greenplum Command Center is an optional web-based performance monitoring and management tool for Greenplum Database, based on the gpperfmon database. Administrators can install Command Center separately from Greenplum Database.

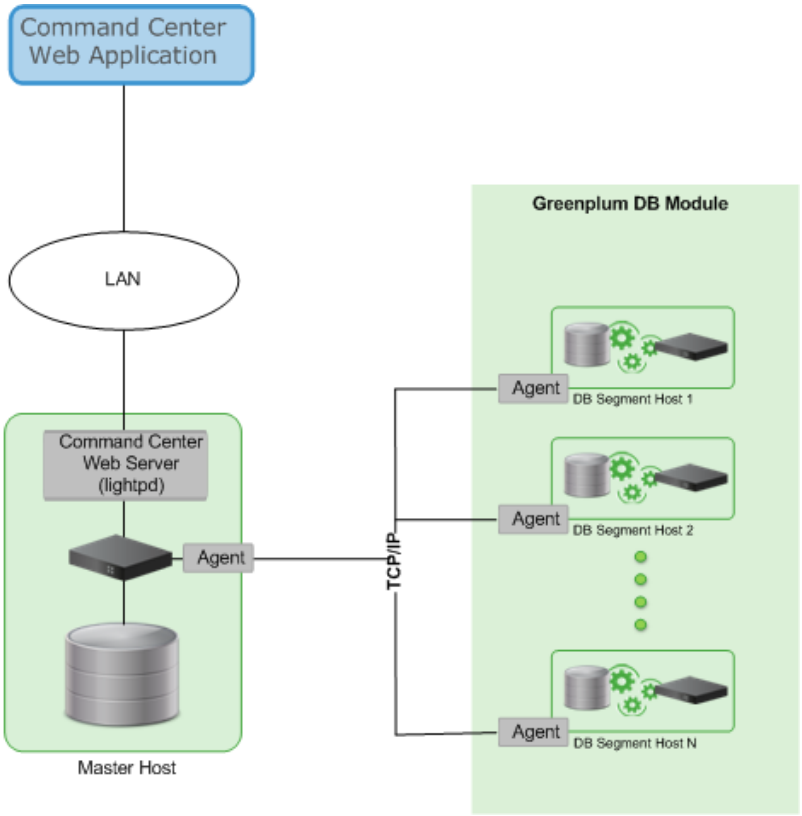


Figure 9: Greenplum Performance Monitoring Architecture

Chapter 3

Estimating Storage Capacity

To estimate how much data your Greenplum Database system can accommodate, use the following measurements as guidelines. Also keep in mind that you may want to have extra space for landing backup files and data load files on each segment host.

Calculating Usable Disk Capacity

To calculate how much data a Greenplum Database system can hold, you have to calculate the usable disk capacity per segment host and then multiply that by the number of segment hosts in your Greenplum Database array. Start with the raw capacity of the physical disks on a segment host that are available for data storage (*raw_capacity*), which is:

```
disk_size * number_of_disks
```

Account for file system formatting overhead (roughly 10 percent) and the RAID level you are using. For example, if using RAID-10, the calculation would be:

```
(raw_capacity * 0.9) / 2 = formatted_disk_space
```

For optimal performance, do not completely fill your disks to capacity, but run at 70% or lower. So with this in mind, calculate the usable disk space as follows:

```
formatted_disk_space * 0.7 = usable_disk_space
```

Once you have formatted RAID disk arrays and accounted for the maximum recommended capacity (*usable_disk_space*), you will need to calculate how much storage is actually available for user data (`U`). If using Greenplum Database mirrors for data redundancy, this would then double the size of your user data (`2 * U`). Greenplum Database also requires some space be reserved as a working area for active queries. The work space should be approximately one third the size of your user data (work space = `U/3`):

```
With mirrors: (2 * U) + U/3 = usable_disk_space

Without mirrors: U + U/3 = usable_disk_space
```

Guidelines for temporary file space and user data space assume a typical analytic workload. Highly concurrent workloads or workloads with queries that require very large amounts of temporary space can benefit from reserving a larger working area. Typically, overall system throughput can be increased while decreasing work area usage through proper workload management. Additionally, temporary space and user space can be isolated from each other by specifying that they reside on different tablespaces.

In the *Greenplum Database Administrator Guide*, see these topics:

- "Managing Workload and Resources" for information about workload management
- "Creating and Managing Tablespaces" for information about moving the location of temporary files
- "Monitoring System State" for information about monitoring Greenplum Database disk space usage

Calculating User Data Size

As with all databases, the size of your raw data will be slightly larger once it is loaded into the database. On average, raw data will be about 1.4 times larger on disk after it is loaded into the database, but could be smaller or larger depending on the data types you are using, table storage type, in-database compression, and so on.

- Page Overhead - When your data is loaded into Greenplum Database, it is divided into pages of 32KB each. Each page has 20 bytes of page overhead.
- Row Overhead - In a regular 'heap' storage table, each row of data has 24 bytes of row overhead. An 'append-optimized' storage table has only 4 bytes of row overhead.
- Attribute Overhead - For the data values itself, the size associated with each attribute value is dependent upon the data type chosen. As a general rule, you want to use the smallest data type possible to store your data (assuming you know the possible values a column will have).
- Indexes - In Greenplum Database, indexes are distributed across the segment hosts as is table data. The default index type in Greenplum Database is B-tree. Because index size depends on the number of unique values in the index and the data to be inserted, precalculating the exact size of an index is impossible. However, you can roughly estimate the size of an index using these formulas.

```
B-tree: unique_values * (data_type_size + 24 bytes)

Bitmap: (unique_values * number_of_rows * 1 bit * compression_ratio / 8) +
 (unique_values * 32)
```

Calculating Space Requirements for Metadata and Logs

On each segment host, you will also want to account for space for Greenplum Database log files and metadata:

- **System Metadata** — For each Greenplum Database segment instance (primary or mirror) or master instance running on a host, estimate approximately 20 MB for the system catalogs and metadata.
- **Write Ahead Log** — For each Greenplum Database segment (primary or mirror) or master instance running on a host, allocate space for the write ahead log (WAL). The WAL is divided into segment files of 64 MB each. At most, the number of WAL files will be:

  ```
  2 * checkpoint_segments + 1
  ```

  You can use this to estimate space requirements for WAL. The default *checkpoint_segments* setting for a Greenplum Database instance is 8, meaning 1088 MB WAL space allocated for each segment or master instance on a host.
- **Greenplum Database Log Files** — Each segment instance and the master instance generates database log files, which will grow over time. Sufficient space should be allocated for these log files, and some type of log rotation facility should be used to ensure that to log files do not grow too large.
- **Command Center Data** — The data collection agents utilized by Command Center run on the same set of hosts as your Greenplum Database instance and utilize the system resources of those hosts. The resource consumption of the data collection agent processes on these hosts is minimal and should not significantly impact database performance. Historical data collected by the collection agents is stored in its own Command Center database (named `gpperfmon`) within your Greenplum Database system. Collected data is distributed just like regular database data, so you will need to account for disk space in the data directory locations of your Greenplum segment instances. The amount of space required depends on the amount of historical data you would like to keep. Historical data is not automatically truncated. Database administrators must set up a truncation policy to maintain the size of the Command Center database.

Chapter 4

Configuring Your Systems and Installing Greenplum

This chapter describes how to prepare your operating system environment for Greenplum, and install the Greenplum Database software binaries on all of the hosts that will comprise your Greenplum Database system. Perform the following tasks in order:

1. Make sure your systems meet the *System Requirements*
2. *Setting the Greenplum Recommended OS Parameters*
3. (master only) *Running the Greenplum Installer*
4. *Installing and Configuring Greenplum on all Hosts*
5. (Optional) *Installing Oracle Compatibility Functions*
6. (Optional) *Installing Greenplum Database Extensions*
7. *Creating the Data Storage Areas*
8. *Synchronizing System Clocks*
9. *Next Steps*

Unless noted, these tasks should be performed for *all* hosts in your Greenplum Database array (master, standby master and segments).

You can install and configure Greenplum Database on virtual servers provided by the Amazon Elastic Compute Cloud (Amazon EC2) web service. For information about using Greenplum Database in an Amazon EC2 environment, see *Amazon EC2 Configuration*.

> **Important:** When data loss is not acceptable for a Pivotal Greenplum Database cluster, master and segment mirroring must be enabled in order for the cluster to be supported by Pivotal. Without mirroring, system and data availability is not guaranteed, Pivotal will make best efforts to restore a cluster in this case. For information about master and segment mirroring, see *About Redundancy and Failover* in the *Greenplum Database Administrator Guide*.
>
> **Note:** For information about upgrading Pivotal Greenplum Database from a previous version, see the *Greenplum Database Release Notes* for the release that you are installing.

System Requirements

The following table lists minimum recommended specifications for servers intended to support Greenplum Database on Linux systems in a production environment. All servers in your Greenplum Database system must have the same hardware and software configuration. Greenplum also provides hardware build guides for its certified hardware platforms. It is recommended that you work with a Greenplum Systems Engineer to review your anticipated environment to ensure an appropriate hardware configuration for Greenplum Database.

Table 3: System Prerequisites for Greenplum Database 4.3

Operating System	**Note:** See the *Greenplum Database Release Notes* for current supported platform information.
File Systems	• xfs required for data storage on SUSE Linux and Red Hat (ext3 supported for root file system)
Minimum CPU	Pentium Pro compatible (P3/Athlon and above)
Minimum Memory	16 GB RAM per server
Disk Requirements	• 150MB per host for Greenplum installation • Approximately 300MB per segment instance for meta data • Appropriate free space for data with disks at no more than 70% capacity • High-speed, local storage
Network Requirements	10 Gigabit Ethernet within the array Dedicated, non-blocking switch NIC bonding is recommended when multiple interfaces are present
Software and Utilities	bash shell GNU tar GNU zip GNU sed (used by Greenplum Database `gpinitsystem`) perl

Important: SSL is supported only on the Greenplum Database master host system. It is not supported on the segment host systems.

Important: For all Greenplum Database host systems, the SELinux must be disabled. You should also disable firewall software such as `iptables` (on systems such as RHEL 6.x and CentOS 6.x) or `firewalld` (on systems such as RHEL 7.x and CentOS 7.x). You can enable firewall software if it is required for security purposes. For information about enabling and configuring `iptables`, see *Enabling iptables*. For information about configuring `firewalld` see your operating system documentation. For information about configuring other firewall software, see the documentation for your firewall software.

This information describes how to check the status and disable SELinux, `iptables`, and `firewalld` features.

- This command checks the status of SELinux when run as root:

  ```
  # sestatus
  SELinuxstatus: disabled
  ```

  You can disable SELinux by editing the `/etc/selinux/config` file. As root, change the value of the `SELINUX` parameter in the `config` file and reboot the system:

  ```
  SELINUX=disabled
  ```

- This command checks the status of `iptables` when run as root:

  ```
  # /sbin/chkconfig --list iptables
  ```

  This is the output if `iptables` is disabled.

  ```
  iptables 0:off 1:off 2:off 3:off 4:off 5:off 6:off
  ```

  One method of disabling `iptables` is to become root, run this command, and then reboot the system:

  ```
  /sbin/chkconfig iptables off
  ```

- This command checks the status of `firewalld` when run as root:

  ```
  # systemctl status firewalld.service
  ```

  This is the output if `firewalld` is disabled.

  ```
  * firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled;
   vendor preset: enabled)
     Active: inactive (dead)
  ```

  These commands disable `firewalld` when run as root:

  ```
  # systemctl stop firewalld.service
  # systemctl disable firewalld.service
  ```

Setting the Greenplum Recommended OS Parameters

Greenplum requires the certain Linux operating system (OS) parameters be set on all hosts in your Greenplum Database system (masters and segments).

In general, the following categories of system parameters need to be altered:

- **Shared Memory** - A Greenplum Database instance will not work unless the shared memory segment for your kernel is properly sized. Most default OS installations have the shared memory values set too low for Greenplum Database. On Linux systems, you must also disable the OOM (out of memory) killer. For information about Greenplum Database shared memory requirements, see the Greenplum Database server configuration parameter `shared_buffers` in the *Greenplum Database Reference Guide*.
- **Network** - On high-volume Greenplum Database systems, certain network-related tuning parameters must be set to optimize network connections made by the Greenplum interconnect.
- **User Limits** - User limits control the resources available to processes started by a user's shell. Greenplum Database requires a higher limit on the allowed number of file descriptors that a single process can have open. The default settings may cause some Greenplum Database queries to fail because they will run out of file descriptors needed to process the query.

Linux System Settings

- Edit the `/etc/hosts` file and make sure that it includes the host names and all interface address names for every machine participating in your Greenplum Database system.
- Set the following parameters in the `/etc/sysctl.conf` file and reboot:

```
kernel.shmmax = 500000000
kernel.shmmni = 4096
kernel.shmall = 4000000000
kernel.sem = 500 1024000 200 4096
kernel.sysrq = 1
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.msgmni = 2048
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.conf.all.arp_filter = 1
net.ipv4.ip_local_port_range = 10000 65535
net.core.netdev_max_backlog = 10000
net.core.rmem_max = 2097152
net.core.wmem_max = 2097152
vm.overcommit_memory = 2
```

Note: Azure deployments require Greenplum to not use port 65330. Add the following line to `sysctl.conf`:

```
net.ipv4.ip_local_reserved_ports=65330
```

Note: When `vm.overcommit_memory` is 2, you specify a value for `vm.overcommit_ratio`. For information about calculating the value for `vm.overcommit_ratio`, see the Greenplum Database server configuration parameter `gp_vmem_protect_limit` in the *Greenplum Database Reference Guide*.

To avoid port conflicts between Greenplum Database and other applications, when initializing Greenplum Database, do not specify Greenplum Database ports in the range specified by the operating system parameter `net.ipv4.ip_local_port_range`. For example, if

`net.ipv4.ip_local_port_range = 10000 65535`, you could set the Greenplum Database base port numbers to these values.

```
PORT_BASE = 6000
MIRROR_PORT_BASE = 7000
REPLICATION_PORT_BASE = 8000
MIRROR_REPLICATION_PORT_BASE = 9000
```

For information about port ranges that are used by Greenplum Database, see *gpinitsystem*.

- Set the following parameters in the `/etc/security/limits.conf` file:

```
* soft nofile 65536
* hard nofile 65536
* soft nproc 131072
* hard nproc 131072
```

For Red Hat Enterprise Linux (RHEL) 6.x and 7.x and CentOS 6.x and 7.x, parameter values in the `/etc/security/limits.d/`*`NN`*`-nproc.conf` file override the values in the `limits.conf` file. If a parameter value is set in both `conf` files, ensure that the parameter is set properly in the *`NN`*`-nproc.conf` file. The Linux module `pam_limits` sets user limits by reading the values from the `limits.conf` file and then from the *`NN`*`-nproc.conf` file. For information about PAM and user limits, see the documentation on PAM and `pam_limits`.

- Each disk device file should have a read-ahead (`blockdev`) value of 16384.

To verify the read-ahead value of a disk device:

```
# /sbin/blockdev --getra devname
```

For example:

```
# /sbin/blockdev --getra /dev/sdb
```

To set blockdev (read-ahead) on a device:

```
# /sbin/blockdev --setra bytes devname
```

For example:

```
# /sbin/blockdev --setra 16384 /dev/sdb
```

See the manual page (man) for the `blockdev` command for more information about using that command (`man blockdev` opens the man page).

- XFS is the preferred file system on Linux platforms for data storage. The following XFS mount options are recommended:

```
rw,nodev,noatime,nobarrier,inode64
```

For RHEL 5.x, CentOS 5.x, and SUSE versions 11.3 and earlier, the `allocsize=16m` option is also recommended.

```
rw,nodev,noatime,nobarrier,inode64,allocsize=16m
```

See the manual page (man) for the `mount` command for more information about using that command (`man mount` opens the man page).

The XFS options can also be set in the `/etc/fstab` file. This example entry from an `fstab` file of a CentOS 7 system specifies the XFS options.

```
/dev/data /data xfs nodev,noatime,nobarrier,inode64 0 0
```

This example entry from an `fstab` file includes the `allocsize=16m` option for RHEL 5.x, CentOS 5.x, and SUSE versions 11.3 and earlier.

```
/dev/data /data xfs nodev,noatime,nobarrier,inode64,allocsize=16m 0 0
```

- The Linux disk I/O scheduler for disk access supports different policies, such as `CFQ`, `AS`, and `deadline`.

  The `deadline` scheduler option is recommended. To specify a scheduler until the next system reboot, run the following:

  ```
  # echo schedulername > /sys/block/devname/queue/scheduler
  ```

  For example:

  ```
  # echo deadline > /sys/block/sbd/queue/scheduler
  ```

  You can specify the I/O scheduler at boot time with the `elevator` kernel parameter. Add the parameter `elevator=deadline` to the `kernel` command in the file `/boot/grub/grub.conf`, the GRUB boot loader configuration file. This is an example `kernel` command from a `grub.conf` file on RHEL 6.x or CentOS 6.x. The command is on multiple lines for readability.

  ```
  kernel /vmlinuz-2.6.18-274.3.1.el5 ro root=LABEL=/
      elevator=deadline crashkernel=128M@16M  quiet console=tty1
      console=ttyS1,115200 panic=30 transparent_hugepage=never
      initrd /initrd-2.6.18-274.3.1.el5.img
  ```

  To specify the I/O scheduler at boot time on systems that use `grub2` such as RHEL 7.x or CentOS 7.x, use the system utility `grubby`. This command adds the parameter when run as root.

  ```
  # grubby --update-kernel=ALL --args="elevator=deadline"
  ```

  After adding the parameter, reboot the system.

  This `grubby` command displays kernel parameter settings.

  ```
  # grubby --info=ALL
  ```

  For more information about the `grubby` utility, see your operating system documentation. If the `grubby` command does not update the kernels, see the *Note* at the end of the section.
- Disable Transparent Huge Pages (THP). RHEL 6.0 or higher enables THP by default. THP degrades Greenplum Database performance. One way to disable THP on RHEL 6.x is by adding the parameter `transparent_hugepage=never` to the `kernel` command in the file `/boot/grub/grub.conf`, the GRUB boot loader configuration file. This is an example `kernel` command from a `grub.conf` file. The command is on multiple lines for readability:

  ```
  kernel /vmlinuz-2.6.18-274.3.1.el5 ro root=LABEL=/
      elevator=deadline crashkernel=128M@16M  quiet console=tty1
      console=ttyS1,115200 panic=30 transparent_hugepage=never
      initrd /initrd-2.6.18-274.3.1.el5.img
  ```

  On systems that use `grub2` such as RHEL 7.x or CentOS 7.x, use the system utility `grubby`. This command adds the parameter when run as root.

  ```
  # grubby --update-kernel=ALL --args="transparent_hugepage=never"
  ```

  After adding the parameter, reboot the system.

  This `cat` command checks the state of THP. The output indicates that THP is disabled.

  ```
  $ cat /sys/kernel/mm/*transparent_hugepage/enabled
  ```

```
always [never]
```

This `grubby` command displays kernel parameter settings.

```
# grubby --info=ALL
```

For more information about Transparent Huge Pages or the `grubby` utility, see your operating system documentation. If the `grubby` command does not update the kernels, see the *Note* at the end of the section.

- Disable IPC object removal for RHEL 7.2 or CentOS 7.2 releases or higher. The default `systemd` setting `RemoveIPC=yes` removes IPC connections when non-system users log out. This causes the Greenplum Database utility `gpinitsystem` to fail with semaphore errors. Perform one of the following to avoid this issue.
  - Create the `gpadmin` user as a system account. For the `useradd` command, the `-r` option creates a user as a system user and the `-m` option creates a home directory for the user.

    Note: When you run the `gpseginstall` utility as the `root` user to install Greenplum Database on host systems, the utility creates the `gpadmin` user as a system account on the hosts. See *Installing and Configuring Greenplum on all Hosts*
  - Disable `RemoveIPC`. Set this parameter in `/etc/systemd/logind.conf` on the Greenplum Database host systems.

    ```
    RemoveIPC=no
    ```

    The setting takes effect after restarting the `systemd-login` service or rebooting the system. To restart the service, run this command as the root user.

    ```
    service systemd-logind restart
    ```
- Certain Greenplum Database management utilities including `gpexpand`, `gpinitsystem`, and `gpaddmirrors`, utilize secure shell (SSH) connections between systems to perform their tasks. In large Greenplum Database deployments, cloud deployments, or deployments with a large number of segments per host, these utilities may exceed the host's maximum threshold for unauthenticated connections. When this occurs, you receive errors such as: `ssh_exchange_identification: Connection closed by remote host.`.

  To increase this connection threshold for your Greenplum Database deployment, update the SSH `MaxStartups` configuration parameter in one of the `/etc/ssh/sshd_config` or `/etc/sshd_config` SSH daemon configuration files.

  If you specify `MaxStartups` using a single integer value, you identify the maximum number of concurrent unauthenticated connections. For example:

  ```
  MaxStartups 200
  ```

  If you specify `MaxStartups` using the "*start:rate:full*" syntax, you enable random early connection drop by the SSH daemon. *start* identifies the maximum number of unathenticated SSH connection attempts allowed. Once *start* number of unauthenticated connection attempts is reached, the SSH daemon refuses *rate* percent of subsequent connection attempts. *full* identifies the maximum number of unauthenticated connection attempts after which all attempts are refused. For example:

  ```
  Max Startups 10:30:200
  ```

  Restart the SSH daemon after you update `MaxStartups`. For example, on a CentOS 6 system, run the following command as the `root` user:

  ```
  # service sshd restart
  ```

For detailed information about SSH configuration options, refer to the SSH documentation for your Linux distribution.

- On some SUSE Linux Enterprise Server platforms, the Greenplum Database utility `gpssh` fails with the error message `out of pty devices`. A workaround is to add Greenplum Database users, for example `gpadmin`, to the tty group. On SUSE systems, tty is required to run `gpssh`.

  Note: If the `grubby` command does not update the kernels of a RHEL 7.x or CentOS 7.x system, you can manually update all kernels on the system. For example, to add the parameter `transparent_hugepage=never` to all kernels on a system.

  1. Add the parameter to the `GRUB_CMDLINE_LINUX` line in the file parameter in `/etc/default/grub`.

     ```
     GRUB_TIMEOUT=5
     GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
     GRUB_DEFAULT=saved
     GRUB_DISABLE_SUBMENU=true
     GRUB_TERMINAL_OUTPUT="console"
     GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=cl/root rd.lvm.lv=cl/swap
      rhgb quiet transparent_hugepage=never"
     GRUB_DISABLE_RECOVERY="true"
     ```

  2. As root, run the `grub2-mkconfig` command to update the kernels.

     ```
     # grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

  3. Reboot the system.

Running the Greenplum Installer

To configure your systems for Greenplum Database, you will need certain utilities found in `$GPHOME/bin` of your installation. Log in as `root` and run the Greenplum installer on the machine that will be your master host.

> **Important:** If you are upgrading Greenplum Database, uninstall the Greenplum Database the gNet extension package if it is installed.
>
> The gNet extension package contains the software for the gphdfs protocol. For Greenplum Database 4.3.1 and later releases, the extension is bundled with Greenplum Database. The files for gphdfs are installed in `$GPHOME/lib/hadoop`.

To install the Greenplum binaries on the master host

1. Download or copy the installer file to the machine that will be the Greenplum Database master host. Installer files are available from Greenplum for Red Hat (32-bit and 64-bit), and SuSe Linux 64-bit platforms.
2. Unzip the installer file where *PLATFORM* is either `RHEL5-i386` (Red Hat 32-bit), `RHEL5-x86_64` (Red Hat 64-bit), or `SuSE10-x86_64` (SuSe Linux 64 bit). For example:

   ```
   # unzip greenplum-db-4.3.x.x-PLATFORM.zip
   ```

3. Launch the installer using `bash`. For example:

   ```
   # /bin/bash greenplum-db-4.3.x.x-PLATFORM.bin
   ```

4. The installer will prompt you to accept the Greenplum Database license agreement. Type `yes` to accept the license agreement.
5. The installer will prompt you to provide an installation path. Press `ENTER` to accept the default install path (`/usr/local/greenplum-db-4.3.x.x`), or enter an absolute path to an install location. You must have write permissions to the location you specify.
6. Optional. The installer will prompt you to provide the path to a previous installation of Greenplum Database. For example: `/usr/local/greenplum-db-4.3.x.x`)

   This installation step will migrate any Greenplum Database add-on modules (`postgis`, `pgcrypto`, etc.) from the previous installation path to the path to the version currently being installed. This step is optional and can be performed manually at any point after the installation using the `gppkg` utility with the `-migrate` option. See *gppkg* for details.

   Press `ENTER` to skip this step.
7. The installer will install the Greenplum software and create a `greenplum-db` symbolic link one directory level above your version-specific Greenplum installation directory. The symbolic link is used to facilitate patch maintenance and upgrades between versions. The installed location is referred to as `$GPHOME`.
8. To perform additional required system configuration tasks and to install Greenplum Database on other hosts, go to the next task *Installing and Configuring Greenplum on all Hosts*.

About Your Greenplum Database Installation

- `greenplum_path.sh` — This file contains the environment variables for Greenplum Database. See *Setting Greenplum Environment Variables*.
- `GPDB-LICENSE.txt` — Greenplum Database license agreement.
- `LICENSE.thirdparty` — License agreements for third-party software included with Greenplum Database.

- **bin** — This directory contains the Greenplum Data`LICENSE-thirdparty`base management utilities. This directory also contains the PostgreSQL client and server programs, most of which are also used in Greenplum Database.
- **demo** — This directory contains the Greenplum MapReduce software.
- **docs/cli_help** — This directory contains help files for Greenplum Database command-line utilities.
- **docs/cli_help/gpconfigs** — This directory contains sample `gpinitsystem` configuration files and host files that can be modified and used when installing and initializing a Greenplum Database system.
- **docs/contrib** — This directory contains copyright, installation, and readme files for the Oracle compatibility functions. See *Installing Oracle Compatibility Functions* for help setting up and using the Oracle compatibility functions with Greenplum Database.
- **docs/javadoc** — This directory contains javadocs for the gNet extension (gphdfs protocol). The jar files for the gNet extension are installed in the `$GPHOME/lib/hadoop` directory.
- **etc** — Sample configuration file for OpenSSL and a sample configuration file to be used with the `gpcheck` management utility.
- **ext** — Bundled programs (such as Python) used by some Greenplum Database utilities.
- **include** — The C header files for Greenplum Database.
- **lib** — Greenplum Database and PostgreSQL library files.
- **sbin** — Supporting/Internal scripts and programs.
- **share** — Shared files for Greenplum Database.

Installing and Configuring Greenplum on all Hosts

When run as `root`, *`gpseginstall`* copies the Greenplum Database installation from the current host and installs it on a list of specified hosts, creates the Greenplum system user (`gpadmin`), sets the system user's password (default is `changeme`), sets the ownership of the Greenplum Database installation directory, and exchanges ssh keys between all specified host address names (both as `root` and as the specified system user).

About gpadmin

When a Greenplum Database system is first initialized, the system contains one predefined superuser role (also referred to as the system user), `gpadmin`. This is the user who owns and administers the Greenplum Database.

> **Note:** If you are setting up a single node system, you can still use *`gpseginstall`* to perform the required system configuration tasks on the current host. In this case, the `hostfile_exkeys` would just have the current host name only.

To install and configure Greenplum Database on all specified hosts

1. Log in to the master host as `root`:

```
$ su -
```

2. Source the path file from your master host's Greenplum Database installation directory:

```
# source /usr/local/greenplum-db/greenplum_path.sh
```

3. Create a file called `hostfile_exkeys` that has the machine configured host names and host addresses (interface names) for each host in your Greenplum system (master, standby master and segments). Make sure there are no blank lines or extra spaces. For example, if you have a master, standby master and three segments with two unbonded network interfaces per host, your file would look something like this:

```
mdw
mdw-1
mdw-2
smdw
smdw-1
smdw-2
sdw1
sdw1-1
sdw1-2
sdw2
sdw2-1
sdw2-2
sdw3
sdw3-1
sdw3-2
```

Check the `/etc/hosts` file on your systems for the correct host names to use for your environment.

The Greenplum Database segment host naming convention is *sdwN* where *sdw* is a prefix and *N* is an integer. For example, on a Greenplum Database DCA system, segment host names would be `sdw1`, `sdw2` and so on. NIC bonding is recommended for hosts with multiple interfaces, but when the interfaces are not bonded, the convention is to append a dash (`-`) and number to the host name. For example, `sdw1-1` and `sdw1-2` are the two interface names for host `sdw1`.

4. Run the *gpseginstall* utility referencing the `hostfile_exkeys` file you just created. This example runs the utility as `root`. The utility creates the Greenplum system user `gpadmin` on all hosts and sets the password as `changeme` for that user on all hosts.

   ```
   # gpseginstall -f hostfile_exkeys
   ```

   Use the `-u` and `-p` options to specify a different system user and password. See *gpseginstall* for option information and running the utility as a non-root user.

Recommended security best practices:

- Do not use the default password option for production environments.
- Change the password immediately after installation.

Confirming Your Installation

To make sure the Greenplum software was installed and configured correctly, run the following confirmation steps from your Greenplum master host. If necessary, correct any problems before continuing on to the next task.

1. Log in to the master host as `gpadmin`:

   ```
   $ su - gpadmin
   ```

2. Source the path file from Greenplum Database installation directory:

   ```
   # source /usr/local/greenplum-db/greenplum_path.sh
   ```

3. Use the `gpssh` utility to see if you can login to all hosts without a password prompt, and to confirm that the Greenplum software was installed on all hosts. Use the `hostfile_exkeys` file you used for installation. For example:

   ```
   $ gpssh -f hostfile_exkeys -e ls -l $GPHOME
   ```

   If the installation was successful, you should be able to log in to all hosts without a password prompt. All hosts should show that they have the same contents in their installation directories, and that the directories are owned by the `gpadmin` user.

   If you are prompted for a password, run the following command to redo the ssh key exchange:

   ```
   $ gpssh-exkeys -f hostfile_exkeys
   ```

Installing Oracle Compatibility Functions

Optional. Many Oracle Compatibility SQL functions are available in Greenplum Database. These functions target PostgreSQL.

Before using any Oracle Compatibility Functions, you need to run the installation script `$GPHOME/share/postgresql/contrib/orafunc.sql` once for each database. For example, to install the functions in database `testdb`, use the command

```
$ psql -d testdb -f $GPHOME/share/postgresql/contrib/orafunc.sql
```

To uninstall Oracle Compatibility Functions, use the script:

```
$GPHOME/share/postgresql/contrib/uninstall_orafunc.sql
```

> **Note:** The following functions are available by default and can be accessed without running the Oracle Compatibility installer: `sinh`, `tanh`, `cosh` and `decode`.

For more information about Greenplum's Oracle compatibility functions, see "Oracle Compatibility Functions" in the *Greenplum Database Utility Guide*.

Installing Greenplum Database Extensions

Optional. Use the Greenplum package manager (`gppkg`) to install Greenplum Database extensions such as pgcrypto, PL/R, PL/Java, PL/Perl, and PostGIS, along with their dependencies, across an entire cluster. The package manager also integrates with existing scripts so that any packages are automatically installed on any new hosts introduced into the system following cluster expansion or segment host recovery.

See *`gppkg`* for more information, including usage.

Extension packages can be downloaded from the Greenplum Database page on *Pivotal Network*. The extension documentation in the *Greenplum Database Reference Guide* contains information about installing extension packages and using extensions.

> **Important:** If you intend to use an extension package with Greenplum Database 4.3.5.0, you must install and use a Greenplum Database extension packages (gppkg files and contrib modules) that are built for Greenplum Database 4.3.5.0. For custom modules that were used with Greenplum Database 4.3.4.x and earlier, you must rebuild the modules for use with Greenplum Database 4.3.5.0.
>
> **Note:** Pivotal supplies separate PL/Perl extension packages for Red Hat Enterprise Linux 7.x, 6.x, and 5.x. Ensure you install the correct package for your operating system.

Creating the Data Storage Areas

Every Greenplum Database master and segment instance has a designated storage area on disk that is called the *data directory* location. This is the file system location where the directories that store segment instance data will be created. The master host needs a data storage location for the master data directory. Each segment host needs a data directory storage location for its primary segments, and another for its mirror segments.

Creating a Data Storage Area on the Master Host

A data storage area is required on the Greenplum Database master host to store Greenplum Database system data such as catalog data and other system metadata.

To create the data directory location on the master

The data directory location on the master is different than those on the segments. The master does not store any user data, only the system catalog tables and system metadata are stored on the master instance, therefore you do not need to designate as much storage space as on the segments.

1. Create or choose a directory that will serve as your master data storage area. This directory should have sufficient disk space for your data and be owned by the `gpadmin` user and group. For example, run the following commands as `root`:

```
# mkdir /data/master
```

2. Change ownership of this directory to the `gpadmin` user. For example:

```
# chown gpadmin /data/master
```

3. Using *gpssh*, create the master data directory location on your standby master as well. For example:

```
# source /usr/local/greenplum-db-4.3.x.x/greenplum_path.sh
# gpssh -h smdw -e 'mkdir /data/master'
# gpssh -h smdw -e 'chown gpadmin /data/master'
```

Creating Data Storage Areas on Segment Hosts

Data storage areas are required on the Greenplum Database segment hosts for primary segments. Separate storage areas are required for mirror segments.

To create the data directory locations on all segment hosts

1. On the master host, log in as `root`:

```
# su
```

2. Create a file called `hostfile_gpssh_segonly`. This file should have only one machine configured host name for each segment host. For example, if you have three segment hosts:

```
sdw1
sdw2
sdw3
```

3. Using *gpssh*, create the primary and mirror data directory locations on all segment hosts at once using the `hostfile_gpssh_segonly` file you just created. For example:

```
# source /usr/local/greenplum-db-4.3.x.x/greenplum_path.sh
# gpssh -f hostfile_gpssh_segonly -e 'mkdir /data/primary'
```

```
# gpssh -f hostfile_gpssh_segonly -e 'mkdir /data/mirror'
# gpssh -f hostfile_gpssh_segonly -e 'chown gpadmin /data/primary'
# gpssh -f hostfile_gpssh_segonly -e 'chown gpadmin /data/mirror'
```

Synchronizing System Clocks

You should use NTP (Network Time Protocol) to synchronize the system clocks on all hosts that comprise your Greenplum Database system. See *www.ntp.org* for more information about NTP.

NTP on the segment hosts should be configured to use the master host as the primary time source, and the standby master as the secondary time source. On the master and standby master hosts, configure NTP to point to your preferred time server.

To configure NTP

1. On the master host, log in as root and edit the `/etc/ntp.conf` file. Set the `server` parameter to point to your data center's NTP time server. For example (if `10.6.220.20` was the IP address of your data center's NTP server):

   ```
   server 10.6.220.20
   ```

2. On each segment host, log in as root and edit the `/etc/ntp.conf` file. Set the first `server` parameter to point to the master host, and the second server parameter to point to the standby master host. For example:

   ```
   server mdw prefer
   server smdw
   ```

3. On the standby master host, log in as root and edit the `/etc/ntp.conf` file. Set the first `server` parameter to point to the primary master host, and the second server parameter to point to your data center's NTP time server. For example:

   ```
   server mdw prefer
   server 10.6.220.20
   ```

4. On the master host, use the NTP daemon synchronize the system clocks on all Greenplum hosts. For example, using *gpssh*:

   ```
   # gpssh -f hostfile_gpssh_allhosts -v -e 'ntpd'
   ```

Enabling iptables

On Linux systems, you can configure and enable the `iptables` firewall to work with Greenplum Database.

> **Note:** Greenplum Database performance might be impacted when `iptables` is enabled. You should test the performance of your application with `iptables` enabled to ensure that performance is acceptable.

For more information about `iptables` see the `iptables` and firewall documentation for your operating system.

How to Enable iptables

1. As gpadmin, the Greenplum Database administrator, run this command on the Greenplum Database master host to stop Greenplum Database:

   ```
   $ gpstop -a
   ```

2. On the Greenplum Database hosts:

   a. Update the file `/etc/sysconfig/iptables` based on the *Example iptables Rules*.

   b. As root user, run these commands to enable `iptables`:

   ```
   # chkconfig iptables on
   # service iptables start
   ```

3. As gpadmin, run this command on the Greenplum Database master host to start Greenplum Database:

   ```
   $ gpstart -a
   ```

   Warning: After enabling `iptables`, this error in the `/var/log/messages` file indicates that the setting for the `iptables` table is too low and needs to be increased.

   ```
   ip_conntrack: table full, dropping packet.
   ```

   As root user, run this command to view the `iptables` table value:

   ```
   # sysctl net.ipv4.netfilter.ip_conntrack_max
   ```

   The following is the recommended setting to ensure that the Greenplum Database workload does not overflow the `iptables` table. The value might need to be adjusted for your hosts:
   `net.ipv4.netfilter.ip_conntrack_max=6553600`

   You can update `/etc/sysctl.conf` file with the value. For setting values in the file, see *Setting the Greenplum Recommended OS Parameters*.

   To set the value until the next reboots run this command as root.

   ```
   # sysctl net.ipv4.netfilter.ip_conntrack_max=6553600
   ```

Example iptables Rules

When `iptables` is enabled, `iptables` manages the IP communication on the host system based on configuration settings (rules). The example rules are used to configure `iptables` for Greenplum Database master host, standby master host, and segment hosts.

- *Example Master and Standby Master iptables Rules*
- *Example Segment Host iptables Rules*

The two sets of rules account for the different types of communication Greenplum Database expects on the master (primary and standby) and segment hosts. The rules should be added to the `/etc/sysconfig/iptables` file of the Greenplum Database hosts. For Greenplum Database, `iptables` rules should allow the following communication:

- For customer facing communication with the Greenplum Database master, allow at least `postgres` and `28080` (`eth1` interface in the example).
- For Greenplum Database system interconnect, allow communication using `tcp`, `udp`, and `icmp` protocols (`eth4` and `eth5` interfaces in the example).

  The network interfaces that you specify in the `iptables` settings are the interfaces for the Greenplum Database hosts that you list in the *hostfile_gpinitsystem* file. You specify the file when you run the `gpinitsystem` command to intialize a Greenplum Database system. See *Initializing a Greenplum Database System* for information about the *hostfile_gpinitsystem* file and the `gpinitsystem` command.
- For the administration network on a Greenplum DCA, allow communication using `ssh`, `snmp`, `ntp`, and `icmp` protocols. (`eth0` interface in the example).

In the `iptables` file, each append rule command (lines starting with `-A`) is a single line.

The example rules should be adjusted for your configuration. For example:

- The append command, the `-A` lines and connection parameter `-i` should match the connectors for your hosts.
- the CIDR network mask information for the source parameter `-s` should match the IP addresses for your network.

Example Master and Standby Master iptables Rules

Example `iptables` rules with comments for the `/etc/sysconfig/iptables` file on the Greenplum Database master host and standby master host.

```
*filter
# Following 3 are default rules. If the packet passes through
# the rule set it gets these rule.
# Drop all inbound packets by default.
# Drop all forwarded (routed) packets.
# Let anything outbound go through.
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
# Accept anything on the loopback interface.
-A INPUT -i lo -j ACCEPT
# If a connection has already been established allow the
# remote host packets for the connection to pass through.
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# These rules let all tcp and udp through on the standard
# interconnect IP addresses and on the interconnect interfaces.
# NOTE: gpsyncmaster uses random tcp ports in the range 1025 to 65535
# and Greenplum Database uses random udp ports in the range 1025 to 65535.
-A INPUT -i eth4 -p udp -s 192.0.2.0/22 -j ACCEPT
-A INPUT -i eth5 -p udp -s 198.51.100.0/22 -j ACCEPT
-A INPUT -i eth4 -p tcp -s 192.0.2.0/22 -j ACCEPT --syn -m state --state NEW
-A INPUT -i eth5 -p tcp -s 198.51.100.0/22 -j ACCEPT --syn -m state --state NEW
# Allow snmp connections on the admin network on Greenplum DCA.
-A INPUT -i eth0 -p udp --dport snmp -s 203.0.113.0/21 -j ACCEPT
-A INPUT -i eth0 -p tcp --dport snmp -s 203.0.113.0/21 -j ACCEPT --syn -m state --
state NEW
# Allow udp/tcp ntp connections on the admin network on Greenplum DCA.
-A INPUT -i eth0 -p udp --dport ntp -s 203.0.113.0/21 -j ACCEPT
-A INPUT -i eth0 -p tcp --dport ntp -s 203.0.113.0/21 -j ACCEPT --syn -m state --
state NEW
# Allow ssh on all networks (This rule can be more strict).
-A INPUT -p tcp --dport ssh -j ACCEPT --syn -m state --state NEW
# Allow Greenplum Database on all networks.
-A INPUT -p tcp --dport postgres -j ACCEPT --syn -m state --state NEW
```

```
# Allow Greenplum Command Center on the customer facing network.
-A INPUT -i eth1 -p tcp --dport 28080 -j ACCEPT --syn -m state --state NEW
# Allow ping and any other icmp traffic on the interconnect networks.
-A INPUT -i eth4 -p icmp -s 192.0.2.0/22 -j ACCEPT
-A INPUT -i eth5 -p icmp -s 198.51.100.0/22 -j ACCEPT
# Allow ping only on the admin network on Greenplum DCA.
-A INPUT -i eth0 -p icmp --icmp-type echo-request -s 203.0.113.0/21 -j ACCEPT
# Log an error if a packet passes through the rules to the default
# INPUT rule (a DROP).
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7
COMMIT
```

Example Segment Host iptables Rules

Example `iptables` rules for the `/etc/sysconfig/iptables` file on the Greenplum Database segment hosts. The rules for segment hosts are similar to the master rules with fewer interfaces and fewer `udp` and `tcp` services.

```
*filter
:INPUT DROP
:FORWARD DROP
:OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i eth2 -p udp -s 192.0.2.0/22 -j ACCEPT
-A INPUT -i eth3 -p udp -s 198.51.100.0/22 -j ACCEPT
-A INPUT -i eth2 -p tcp -s 192.0.2.0/22 -j ACCEPT --syn -m state --state NEW
-A INPUT -i eth3 -p tcp -s 198.51.100.0/22 -j ACCEPT --syn -m state --state NEW
-A INPUT -i eth0 -p udp --dport snmp -s 203.0.113.0/21 -j ACCEPT
-A INPUT -i eth0 -p tcp --dport snmp -j ACCEPT --syn -m state --state NEW
-A INPUT -p tcp --dport ssh -j ACCEPT --syn -m state --state NEW
-A INPUT -i eth2 -p icmp -s 192.0.2.0/22 -j ACCEPT
-A INPUT -i eth3 -p icmp -s 198.51.100.0/22 -j ACCEPT
-A INPUT -i eth0 -p icmp --icmp-type echo-request -s 203.0.113.0/21 -j ACCEPT
-A INPUT -m limit --limit 5/min -j LOG --log-prefix "iptables denied: " --log-level 7
COMMIT
```

Amazon EC2 Configuration (Amazon Web Services)

You can install and configure Greenplum Database on virtual servers provided by the Amazon Elastic Compute Cloud (Amazon EC2) web service. Amazon EC2 is a service provided by Amazon Web Services (AWS). The following overview information describes how to install Greenplum Database in an Amazon EC2 environment.

About Amazon EC2

You can use Amazon EC2 to launch as many virtual servers as you need, configure security and networking, and manage storage. An EC2 *instance* is a virtual server in the AWS cloud virtual computing environment.

EC2 instances are manged by AWS. AWS isolates your EC2 instances from other users in a virtual private cloud (VPC) and lets you control access to the instances. You can configure instance features such as operating system, network connectivity (network ports and protocols, IP address access), access to the to the Internet, and size and type of disk storage.

When you launch an instance, you use a preconfigured template for your instance, known as an Amazon Machine Image (AMI). The AMI packages the bits you need for your server (including the operating system and additional software). You can use images supplied by Amazon or use customized images. You launch instances in an Availability Zone of an AWS region. An *Availability Zone* is distinct location within a region that are engineered to be insulated from failures in other Availability Zones.

For information about Amazon EC2, see *http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html*

Launching EC2 Instances with the EC2 Console

You can launch instances, configure, start, stop, and terminate (delete) virtual servers, with the Amazon EC2 Console. When you launch instances, you select these features.

Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications).

For Greenplum Database - Select an AMI that runs a supported operating system. See the *Greenplum Database Release Notes* for the release that you are installing.

Note: You create and launch a customized AMI, see *About Amazon Machine Image (AMI)*

EC2 Instance Type

A predefined set performance characteristics. Instance types comprise varying combinations of CPU, memory, default storage, and networking capacity. You can modify storage options when you add storage.

For Greenplum Database - The instance type must be an EBS-Optimized instance type when using Amazon EBS storage for Greenplum Database. See *Configure storage* for information about Greenplum Database storage requirements. For information about EBS-Optimized instances, see the Amazon documentation about *EBS-Optimized instances*.

For sufficient network performance, the instance type must also support EC2 enhanced networking. For information about EC2 enhanced networking, see the Amazon documentation about *Enhanced Networking on Linux Instances*.

The instances should be in a single VPC and subnet. Instances are always assigned a VPC internal IP address and can be assigned a public IP address for external and Internet access.

The internal IP address is used for Greenplum Database communication between hosts. You can also use the internal IP address to access an instance from another instance within the EC2 VPC. For information about configuring launched instances to communicate with each other, see *Working with EC2 Instances*.

A public IP address for the instance and an Internet gateway configured for the EC2 VPC are required for accessing the instance from an external source and for the instance to access the Internet. Internet access is required when installing Linux packages. When you launch a set of instances, you can enable or disable the automatic assignment of public IP addresses when the instances are started.

If automatic assignment of public IP addresses is enabled, instances are always assigned a public IP address when the instance starts. If automatic assignment of public IP addresses is disabled, you can associate a public IP address with the EC2 elastic IP, and temporarily associate public IP addresses to an instance to connect and configure the instance.

To control whether a public IP is assigned when launching an instance, see the Amazon documentation about *Subnet Public IP Addressing*.

EC2 Instance Details

Information about starting, running, and stopping EC2 instances, such as such as number of instances of the same AMI, network information, and EC2 VPC and subnet membership.

Configure storage

Adjust and add storage. For example, you can change the size of root volume and add volumes.

For Greenplum Database - Greenplum Database supports either EC2 instance store or Amazon EBS storage in a production environment.

- EC2 *instance store* provides temporary block-level storage. This storage is located on disks that are physically attached to the host computer. With instance store, powering off the instance causes data loss. Soft reboots preserve instance store data. However, EC2 instance store can provide higher and more consistent I/O performance.
- *EBS storage* provides block level storage volumes with long-term persistence. EBS storage must be mounted with the XFS file system for it to be a supported configuration. All other file systems are explicitly not supported by Pivotal.

  There are several classes of EBS. For Greenplum Database, select the EBS volume type `gp2` for root and swap partitions and `st1` for data. See the Amazon documentation about *Block Device Mapping*.

For more information about the Amazon storage types, see *Notes*.

Create Tag

An optional label that consists of a case-sensitive key-value pair that is used for organizing searching a large number of EC2 resources.

Security Group

A set of firewall rules that control the network traffic for instances.

For external access to an instance with `ssh`, create a rule that enables `ssh` for inbound network traffic.

Working with EC2 Instances

After the EC2 instances have started, you connect to and configure the instances. The **Instances** page of the EC2 Console lists the running instances and network information. If the instance does not have a public IP address, you can create an Elastic IP and associate it with the instance. See *About Amazon Elastic IP Addresses*.

To access EC2 instances, AWS uses public-key cryptography to secure the login information for your instance. A Linux instance has no password; you use a key pair to log in to your instance securely. You specify the name of the key pair when you launch your instance, then provide the private key when you log in using SSH. See the Amazon documentation about *EC2 Key Pairs*.

A key pair consists of a *public key* that AWS stores, and a *private key file* that you store. Together, they allow you to connect to your instance securely.

This example logs into an into EC2 instance from an external location with the private key file `my-test.pem` and user `ec2-user`. The user `ec2-user` is the default user for some Linux AMI templates. This example assumes that the instance is configured with a public IP address `192.0.2.82` and that the `pem` file is the private key file that is used to access the instance.

```
ssh -i my-test.pem ec2-user@192.0.2.82
```

You can also copy the private key file to your home `.ssh` directory as the `id_rsa` file. This example, creates and configures the `id_rsa` file.

```
cp my-test.pem ~/.ssh/id_rsa
chmod 400 .ssh/id_rsa
```

You can also copy the `id_rsa` to your EC2 instances. This `scp` command copies the file to the `.ssh` directory of the `ec2-user`.

```
scp ~/.ssh/id_rsa ec2-user@192.0.2.86:~/.ssh/id_rsa
```

> **Note:** `gpssh-extkey` is not used with Greenplum Database hosts that are EC2 instances. You must copy the private key file to the `.ssh` directory in the user home directory for each instance.

This example logs into an into EC2 instance using the `id_rsa` file.

```
ssh ec2-user@192.0.2.82
```

After the key file is installed on all Greenplum Database hosts you can use Greenplum Database utilities such as `gpseginstall`, `gpssh`, and `gpscp` that access multiple Greenplum Database hosts.

Before installing Greenplum Database, you configure the EC2 instances as you would a local host server machines. Configure the host operating system, configure host network information (for example, update the `/etc/hosts` file), set operating system parameters, and install operating system packages. For information about how to prepare your operating system environment for Greenplum Database, see *Configuring Your Systems and Installing Greenplum*.

These example commands use yum to install the Linux packages `sed`, `unzip`, and `vim`.

```
sudo yum install -y sed
sudo yum install -y unzip
sudo yum install -y vim
```

This example uploads Greenplum Database install file to an EC2 instance to the `ec2-user` home directory.

```
scp greenplum-db-4.3.10.0-build-1-RHEL5-x86_64.zip ec2-user@192.0.2.82:~/.
```

These example commands log into the instance and run the Greenplum Database install file that is in `ec2-user` home directory as the as `ec2-user`.

```
ssh ec2-user@192.0.2.82
unzip greenplum-db-4.3.10.0-build-1-RHEL5-x86_64.zip
./greenplum-db-4.3.10.0-build-1-RHEL5-x86_64.bin
```

This example command runs the `gpseginstall` utility that specifies the user as `ec2-user`. This example assumes the file `my-hosts` contains the instances that are used as Greenplum Database segment hosts and that the instances have been prepared for Greenplum Database.

```
gpseginstall -u ec2-user -f my-hosts
```

> **Note:** During the Greenplum Database installation process, you might see `ssh` messages to confirm the authenticity of host connections. Enter `yes` to confirm the authenticity.

About Amazon Machine Image (AMI)

An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, you launch an *instance*, which is a copy of the AMI running as a virtual server in the cloud. You can launch multiple instances of an AMI.

After you launch an instance, it acts like a traditional host, and you can interact with it as you would any computer. You have complete control of your instances; you can use `sudo` to run commands that require root privileges.

You can create a customized Amazon EBS-backed Linux AMI from an instance that you've launched from an existing Amazon EBS-backed Linux AMI. After you've customized the instance to suit your needs, create and register a new AMI, which you can use to launch new instances with these customizations.

For information about AMI, see the Amazon documentation about *AMIs*.

About Amazon Elastic IP Addresses

An EC2 Elastic IP address is a public IP address that you can allocate (create) for your account. You can associate it to and disassociate it from instances as you require, and it's allocated to your account until you choose to release it.

Your default VPC is configured with an Internet gateway. When you allocate an EC2 Elastic IP address, AWS configures the VPC to allow internet access to the IP address using the gateway.

To enable an instance in your VPC to communicate with the Internet, it must have a public IP address or an EC2 Elastic IP address that's associated with a private IP address on your instance.

To ensure that your instances can communicate with the Internet, you must also attach an Internet gateway to your EC2 VPC. For information about VPC Internet Gateways, see the Amazon documentation about *Internet gateways*.

For information about EC2 Elastic IP addresses and how to use them, see see the Amazon documentation about *Elastic IP Addresses*.

Notes

- The Greenplum Database utility `gpssh-extkey` is not used with Greenplum Database hosts that are EC2 instances. You must copy the private key file to the `.ssh` directory in the user home directory for each instance.
- When you use Amazon EBS storage for Greenplum Database storage, the storage must be mounted with the XFS file system for it to be a supported configuration.

  For information about EBS storage, see the Amazon documentation about *Amazon EBS*. Also, see the Amazon EC2 documentation for configuring the Amazon EBS volumes and managing storage and file systems used with EC2 instances.
- For an EC2 instance with instance store, the virtual devices for instance store volumes are `ephemeral`*`N`* (n is between 0 and 23). On an instance running CentOS the instance store block device names appear as `/dev/xvd`*`letter`*. A RAID0 configuration is recommended with ephemeral disks because there are more disks than segments on each host.

An example of an EC2 instance type that is configured with instance store and that showed acceptable performance is the `d2.8xlarge` instance type configured with four `raid0` volumes of 6 disks each.

For information about EC2 instance store, see the Amazon documentation about *EC2 Instance Store*.

- These are default ports in a Greenplum Database environment. These ports need to be open in the security group to allow access from a source external to a VPC.

Port	Used by this application
22	ssh - connect to host with ssh
5432	Greenplum Database (master)
28080	Greenplum Command Center

- For a non-default VPC you can configure the VPC with an internet gateway for the VPC and allocate Elastic IP address for the VPC. AWS will automatically configure the Elastic IP for internet access. For information about EC2 internet gateways, see the Amazon documentation about *Internet Gateways*.
- A *placement group* is a logical grouping of instances within a single Availability Zone. Using placement groups enables applications to participate in a low-latency, 10 Gbps network. Placement groups are recommended for applications that benefit from low network latency, high network throughput, or both.

  Placement Groups provide the ability for EC2 instances to separated from other instances. However, configuring instances are in different placement groups can improve performance but might create a configuration where an instance in a placement group cannot be replaced.

  See the Amazon documentation about *Placement Groups*.
- Amazon EC2 provides enhanced networking capabilities using single root I/O virtualization (SR-IOV) on some instance types. Enabling enhanced networking on your instance results in higher performance (packets per second), lower latency, and lower jitter.

  To enable enhanced networking on your Red Hat and CentOS RHEL/CentOS instance, you must ensure that the kernel has the ixgbevf module version 2.14.2 or higher is installed and that the sriovNetSupport attribute is set.

  For information about EC2 enhanced networking, see the Amazon documentation about *Enhanced Networking on Linux Instances*.

References

References to AWS and EC2 features and related information.

- AWS - *https://aws.amazon.com/*.
- Connecting to an EC2 instance - *http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html*.
- Amazon VPC - *http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Introduction.html*.
- Amazon EC2 best practices - *https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-best-practices.html*
- Amazon EC2 and VPC command line interface (CLI) - *http://docs.aws.amazon.com/AWSEC2/latest/CommandLineReference/Welcome.html*.
- Amazon S3 (secure, durable, scalable object storage) - *https://aws.amazon.com/s3/*.
- AWS CloudFormation - *https://aws.amazon.com/cloudformation/*.

  With AWS CloudFormation, you can create templates to simplify provisioning and management of related AWS resources.

Next Steps

After you have configured the operating system environment and installed the Greenplum Database software on all of the hosts in the system, the next steps are:

- *Validating Your Systems*
- *Initializing a Greenplum Database System*

Chapter 5

Installing the Data Science Packages

Information about installing the Greenplum Database Python and R Data Science Packages.

Python Data Science Module Package

Greenplum Database provides a collection of data science-related Python modules that can be used with the Greenplum Database PL/Python language. You can download these modules in `.gppkg` format from *Pivotal Network*.

This section contains the following information:

- *Python Data Science Modules*
- *Installing the Python Data Science Module Package*
- *Uninstalling the Python Data Science Module Package*

For information about the Greenplum Database PL/Python Language, see *Greenplum PL/Python Language Extension* in the *Greenplum Database Reference Guide*.

Python Data Science Modules

Modules provided in the Python Data Science package include:

Table 4: Data Science Modules

Module Name	Description/Used For
Beautiful Soup	Navigating HTML and XML
Gensim	Topic modeling and document indexing
lxml	XML and HTML processing
NumPy	Scientific computing
Pandas	Data analysis
Pattern-en	Part-of-speech tagging
pyLDAvis	Interactive topic model visualization
scikit-learn	Machine learning data mining and analysis
SciPy	Scientific computing
StatsModels	Statistical modeling
XGBoost	Gradient boosting, classifying, ranking

Installing the Python Data Science Module Package

Before you install the Python Data Science Module package, make sure that your Greenplum Database is running, you have sourced `greenplum_path.sh`, and that the `$MASTER_DATA_DIRECTORY` and `$GPHOME` environment variables are set.

> **Note:** The `PyMC3` module depends on `Tk`. If you want to use `PyMC3`, you must install the `tk` OS package on every node in your cluster. For example:
>
> ```
> $ yum install tk
> ```

1. Locate the Python Data Science module package that you built or downloaded.

   The file name format of the package is `DataSciencePython-<version>-rhel<N>-x86_64.gppkg`.
2. Copy the package to the Greenplum Database master host.

3. Use the `gppkg` command to install the package. For example:

```
$ gppkg -i DataSciencePython-<version>-rhel<N>-x86_64.gppkg
```

   `gppkg` installs the Python Data Science modules on all nodes in your Greenplum Database cluster. The command also updates the `PYTHONPATH`, `PATH`, and `LD_LIBRARY_PATH` environment variables in your `greenplum_path.sh` file.
4. Restart Greenplum Database. You must re-source `greenplum_path.sh` before restarting your Greenplum cluster:

```
$ source /usr/local/greenplum-db/greenplum_path.sh
$ gpstop -r
```

The Greenplum Database Python Data Science Modules are installed in the following directory:

```
$GPHOME/ext/DataSciencePython/lib/python2.6/site-packages/
```

Uninstalling the Python Data Science Module Package

Use the `gppkg` utility to uninstall the Python Data Science Module package. You must include the version number in the package name you provide to `gppkg`.

To determine your Python Data Science Module package version number and remove this package:

```
$ gppkg -q --all | grep DataSciencePython
DataSciencePython-<version>
$ gppkg -r DataSciencePython-<version>
```

The command removes the Python Data Science modules from your Greenplum Database cluster. It also updates the `PYTHONPATH`, `PATH`, and `LD_LIBRARY_PATH` environment variables in your `greenplum_path.sh` file to their pre-installation values.

Re-source `greenplum_path.sh` and restart Greenplum Database after you remove the Python Data Science Module package:

```
$ . /usr/local/greenplum-db/greenplum_path.sh
$ gpstop -r
```

> **Note:** When you uninstall the Python Data Science Module package from your Greenplum Database cluster, any UDFs that you have created that import Python modules installed with this package will return an error.

R Data Science Library Package

R packages are modules that contain R functions and data sets. Greenplum Database provides a collection of data science-related R libraries that can be used with the Greenplum Database PL/R language. You can download these libraries in `.gppkg` format from *Pivotal Network*.

This chapter contains the following information:

- *R Data Science Libraries*
- *Installing the R Data Science Library Package*
- *Uninstalling the R Data Science Library Package*

For information about the Greenplum Database PL/R Language, see *Greenplum PL/R Language Extension* in the *Greenplum Database Reference Guide*.

R Data Science Libraries

Libraries provided in the R Data Science package include:

abind	gplots	R2jags
adabag	gtable	R6
arm	gtools	randomForest
assertthat	hms	RColorBrewer
BH	hybridHclust	Rcpp
bitops	igraph	RcppEigen
car	labeling	readr
caret	lattice	reshape2
caTools	lazyeval	rjags
coda	lme4	RobustRankAggreg
colorspace	lmtest	ROCR
compHclust	magrittr	rpart
curl	MASS	RPostgreSQL
data.table	Matrix	sandwich
DBI	MCMCPack	scales
dichromat	minqa	SparseM
digest	MTS	stringi
dplyr	munsell	stringr
e1071	neuralnet	survival
flashClust	nloptr	tibble
foreign	nnet	tseries
gdata	pbkrtest	zoo
ggplot2	plyr	

glmnet	quantreg	

Installing the R Data Science Library Package

Before you install the R Data Science Library package, make sure that your Greenplum Database is running, you have sourced `greenplum_path.sh`, and that the `$MASTER_DATA_DIRECTORY` and `$GPHOME` environment variables are set.

1. Locate the R Data Science library package that you built or downloaded.

   The file name format of the package is `DataScienceR-<version>-relhel<N>-x86_64.gppkg`.
2. Copy the package to the Greenplum Database master host.
3. Use the `gppkg` command to install the package. For example:

   ```
   $ gppkg -i DataScienceR-<version>-relhel<N>-x86_64.gppkg
   ```

   `gppkg` installs the R Data Science libraries on all nodes in your Greenplum Database cluster. The command also sets the `R_LIBS_USER` environment variable and updates the `PATH` and `LD_LIBRARY_PATH` environment variables in your `greenplum_path.sh` file.
4. Restart Greenplum Database. You must re-source `greenplum_path.sh` before restarting your Greenplum cluster:

   ```
   $ source /usr/local/greenplum-db/greenplum_path.sh
   $ gpstop -r
   ```

The Greenplum Database R Data Science Modules are installed in the following directory:

```
$GPHOME/ext/DataScienceR/library
```

> **Note:** `rjags` libraries are installed in the `$GPHOME/ext/DataScienceR/extlib/lib` directory. If you want to use `rjags` and your `$GPHOME` is not `/usr/local/greenplum-db`, you must perform additional configuration steps to create a symbolic link from `$GPHOME` to `/usr/local/greenplum-db` on each node in your Greenplum Database cluster. For example:
>
> ```
> $ gpssh -f all_hosts -e 'ln -s $GPHOME /usr/local/greenplum-db'
> $ gpssh -f all_hosts -e 'chown -h gpadmin /usr/local/greenplum-db'
> ```

Uninstalling the R Data Science Library Package

Use the `gppkg` utility to uninstall the R Data Science Library package. You must include the version number in the package name you provide to `gppkg`.

To determine your R Data Science Library package version number and remove this package:

```
$ gppkg -q --all | grep DataScienceR
DataScienceR-<version>
$ gppkg -r DataScienceR-<version>
```

The command removes the R Data Science libraries from your Greenplum Database cluster. It also removes the `R_LIBS_USER` environment variable and updates the `PATH` and `LD_LIBRARY_PATH` environment variables in your `greenplum_path.sh` file to their pre-installation values.

Re-source `greenplum_path.sh` and restart Greenplum Database after you remove the R Data Science Library package:

```
$ . /usr/local/greenplum-db/greenplum_path.sh
$ gpstop -r
```

Note: When you uninstall the R Data Science Library package from your Greenplum Database cluster, any UDFs that you have created that use R libraries installed with this package will return an error.

Chapter 6

Validating Your Systems

Greenplum provides the following utilities to validate the configuration and performance of your systems:

- *gpcheck*
- *gpcheckperf*

These utilities can be found in `$GPHOME/bin` of your Greenplum installation.

The following tests should be run prior to initializing your Greenplum Database system.

Validating OS Settings

Greenplum provides a utility called *gpcheck* that can be used to verify that all hosts in your array have the recommended OS settings for running a production Greenplum Database system. To run `gpcheck`:

1. Log in on the master host as the `gpadmin` user.
2. Source the `greenplum_path.sh` path file from your Greenplum installation. For example:

```
$ source /usr/local/greenplum-db/greenplum_path.sh
```

3. Create a file called `hostfile_gpcheck` that has the machine-configured host names of each Greenplum host (master, standby master and segments), one host name per line. Make sure there are no blank lines or extra spaces. This file should just have a *single* host name per host. For example:

```
mdw
smdw
sdw1
sdw2
sdw3
```

4. Run the *gpcheck* utility using the host file you just created. For example:

```
$ gpcheck -f hostfile_gpcheck -m mdw -s smdw
```

5. After *gpcheck* finishes verifying OS parameters on all hosts (masters and segments), you might be prompted to modify certain OS parameters before initializing your Greenplum Database system.

Validating Hardware Performance

Greenplum provides a management utility called *gpcheckperf*, which can be used to identify hardware and system-level issues on the machines in your Greenplum Database array. *gpcheckperf* starts a session on the specified hosts and runs the following performance tests:

- Network Performance (`gpnetbench*`)
- Disk I/O Performance (`dd` test)
- Memory Bandwidth (`stream` test)

Before using `gpcheckperf`, you must have a trusted host setup between the hosts involved in the performance test. You can use the utility *gpssh-exkeys* to update the known host files and exchange public keys between hosts if you have not done so already. Note that `gpcheckperf` calls to *gpssh* and *gpscp*, so these Greenplum utilities must be in your `$PATH`.

Validating Network Performance

To test network performance, run *gpcheckperf* with one of the network test run options: parallel pair test (`-r N`), serial pair test (`-r n`), or full matrix test (`-r M`). The utility runs a network benchmark program that transfers a 5 second stream of data from the current host to each remote host included in the test. By default, the data is transferred in parallel to each remote host and the minimum, maximum, average and median network transfer rates are reported in megabytes (MB) per second. If the summary transfer rate is slower than expected (less than 100 MB/s), you can run the network test serially using the `-r n` option to obtain per-host results. To run a full-matrix bandwidth test, you can specify `-r M` which will cause every host to send and receive data from every other host specified. This test is best used to validate if the switch fabric can tolerate a full-matrix workload.

Most systems in a Greenplum Database array are configured with multiple network interface cards (NICs), each NIC on its own subnet. When testing network performance, it is important to test each subnet individually. For example, considering the following network configuration of two NICs per host:

Table 5: Example Network Interface Configuration

Greenplum Host	Subnet1 NICs	Subnet2 NICs
Segment 1	sdw1-1	sdw1-2
Segment 2	sdw2-1	sdw2-2
Segment 3	sdw3-1	sdw3-2

You would create four distinct host files for use with the *gpcheckperf* network test:

Table 6: Example Network Test Host File Contents

hostfile_gpchecknet_ic1	hostfile_gpchecknet_ic2
sdw1-1	sdw1-2
sdw2-1	sdw2-2
sdw3-1	sdw3-2

You would then run *gpcheckperf* once per subnet. For example (if testing an *even* number of hosts, run in parallel pairs test mode):

```
$ gpcheckperf -f hostfile_gpchecknet_ic1 -r N -d /tmp > subnet1.out
```

```
$ gpcheckperf -f hostfile_gpchecknet_ic2 -r N -d /tmp > subnet2.out
```

If you have an *odd* number of hosts to test, you can run in serial test mode (`-r n`).

Validating Disk I/O and Memory Bandwidth

To test disk and memory bandwidth performance, run *gpcheckperf* with the disk and stream test run options (`-r ds`). The disk test uses the `dd` command (a standard UNIX utility) to test the sequential throughput performance of a logical disk or file system. The memory test uses the STREAM benchmark program to measure sustainable memory bandwidth. Results are reported in MB per second (MB/s).

To run the disk and stream tests

1. Log in on the master host as the `gpadmin` user.
2. Source the `greenplum_path.sh` path file from your Greenplum installation. For example:

```
$ source /usr/local/greenplum-db/greenplum_path.sh
```

3. Create a host file named `hostfile_gpcheckperf` that has one host name per segment host. Do not include the master host. For example:

```
sdw1
sdw2
sdw3
sdw4
```

4. Run the `gpcheckperf` utility using the `hostfile_gpcheckperf` file you just created. Use the `-d` option to specify the file systems you want to test on each host (you must have write access to these directories). You will want to test all primary and mirror segment data directory locations. For example:

```
$ gpcheckperf -f hostfile_gpcheckperf -r ds -D \
  -d /data1/primary -d  /data2/primary \
  -d /data1/mirror -d  /data2/mirror
```

5. The utility may take a while to perform the tests as it is copying very large files between the hosts. When it is finished you will see the summary results for the Disk Write, Disk Read, and Stream tests.

Chapter 7

Configuring Localization Settings

This chapter describes the available localization features of Greenplum Database. Greenplum Database supports localization with two approaches:

- Using the locale features of the operating system to provide locale-specific collation order, number formatting, and so on.
- Providing a number of different character sets defined in the Greenplum Database server, including multiple-byte character sets, to support storing text in all kinds of languages, and providing character set translation between client and server.

About Locale Support in Greenplum Database

Locale support refers to an application respecting cultural preferences regarding alphabets, sorting, number formatting, etc. Greenplum Database uses the standard ISO C and POSIX locale facilities provided by the server operating system. For additional information refer to the documentation of your operating system.

Locale support is automatically initialized when a Greenplum Database system is initialized. The initialization utility, *gpinitsystem*, will initialize the Greenplum array with the locale setting of its execution environment by default, so if your system is already set to use the locale that you want in your Greenplum Database system then there is nothing else you need to do.

When you are ready to initiate Greenplum Database and you want to use a different locale (or you are not sure which locale your system is set to), you can instruct `gpinitsystem` exactly which locale to use by specifying the -n *locale* option. For example:

```
$ gpinitsystem -c gp_init_config -n sv_SE
```

See *Initializing a Greenplum Database System* for information about the database initialization process.

The example above sets the locale to Swedish (sv) as spoken in Sweden (SE). Other possibilities might be `en_US` (U.S. English) and `fr_CA` (French Canadian). If more than one character set can be useful for a locale then the specifications look like this: `cs_CZ.ISO8859-2`. What locales are available under what names on your system depends on what was provided by the operating system vendor and what was installed. On most systems, the command `locale -a` will provide a list of available locales.

Occasionally it is useful to mix rules from several locales, for example use English collation rules but Spanish messages. To support that, a set of locale subcategories exist that control only a certain aspect of the localization rules:

- `LC_COLLATE` — String sort order
- `LC_CTYPE` — Character classification (What is a letter? Its upper-case equivalent?)
- `LC_MESSAGES` — Language of messages
- `LC_MONETARY` — Formatting of currency amounts
- `LC_NUMERIC` — Formatting of numbers
- `LC_TIME` — Formatting of dates and times

If you want the system to behave as if it had no locale support, use the special locale `C` or `POSIX`.

The nature of some locale categories is that their value has to be fixed for the lifetime of a Greenplum Database system. That is, once `gpinitsystem` has run, you cannot change them anymore. `LC_COLLATE` and `LC_CTYPE` are those categories. They affect the sort order of indexes, so they must be kept fixed, or indexes on text columns will become corrupt. Greenplum Database enforces this by recording the values of `LC_COLLATE` and `LC_CTYPE` that are seen by gpinitsystem. The server automatically adopts those two values based on the locale that was chosen at initialization time.

The other locale categories can be changed as desired whenever the server is running by setting the server configuration parameters that have the same name as the locale categories (see the *Greenplum Database Reference Guide* for more information on setting server configuration parameters). The defaults that are chosen by gpinitsystem are written into the master and segment `postgresql.conf` configuration files to serve as defaults when the Greenplum Database system is started. If you delete these assignments from the master and each segment `postgresql.conf` files then the server will inherit the settings from its execution environment.

Note that the locale behavior of the server is determined by the environment variables seen by the server, not by the environment of any client. Therefore, be careful to configure the correct locale settings on each Greenplum Database host (master and segments) before starting the system. A consequence of this is that

if client and server are set up in different locales, messages may appear in different languages depending on where they originated.

Inheriting the locale from the execution environment means the following on most operating systems: For a given locale category, say the collation, the following environment variables are consulted in this order until one is found to be set: `LC_ALL`, `LC_COLLATE` (the variable corresponding to the respective category), `LANG`. If none of these environment variables are set then the locale defaults to `C`.

Some message localization libraries also look at the environment variable `LANGUAGE` which overrides all other locale settings for the purpose of setting the language of messages. If in doubt, please refer to the documentation for your operating system, in particular the documentation about `gettext`, for more information.

Native language support (NLS), which enables messages to be translated to the user's preferred language, is not enabled in Greenplum Database for languages other than English. This is independent of the other locale support.

Locale Behavior

The locale settings influence the following SQL features:

- Sort order in queries using `ORDER BY` on textual data
- The ability to use indexes with `LIKE` clauses
- The `upper`, `lower`, and `initcap` functions
- The `to_char` family of functions

The drawback of using locales other than `C` or `POSIX` in Greenplum Database is its performance impact. It slows character handling and prevents ordinary indexes from being used by `LIKE`. For this reason use locales only if you actually need them.

Troubleshooting Locales

If locale support does not work as expected, check that the locale support in your operating system is correctly configured. To check what locales are installed on your system, you may use the command `locale -a` if your operating system provides it.

Check that Greenplum Database is actually using the locale that you think it is. `LC_COLLATE` and `LC_CTYPE` settings are determined at initialization time and cannot be changed without redoing *`gpinitsystem`*. Other locale settings including `LC_MESSAGES` and `LC_MONETARY` are initially determined by the operating system environment of the master and/or segment host, but can be changed after initialization by editing the `postgresql.conf` file of each Greenplum master and segment instance. You can check the active locale settings of the master host using the `SHOW` command. Note that every host in your Greenplum Database array should be using identical locale settings.

Character Set Support

The character set support in Greenplum Database allows you to store text in a variety of character sets, including single-byte character sets such as the ISO 8859 series and multiple-byte character sets such as EUC (Extended Unix Code), UTF-8, and Mule internal code. All supported character sets can be used transparently by clients, but a few are not supported for use within the server (that is, as a server-side encoding). The default character set is selected while initializing your Greenplum Database array using `gpinitsystem`. It can be overridden when you create a database, so you can have multiple databases each with a different character set.

Table 7: Greenplum Database Character Sets

Name	Description	Language	Server?	Bytes/Char	Aliases
BIG5	Big Five	Traditional Chinese	No	1-2	WIN950, Windows950
EUC_CN	Extended UNIX Code-CN	Simplified Chinese	Yes	1-3	
EUC_JP	Extended UNIX Code-JP	Japanese	Yes	1-3	
EUC_KR	Extended UNIX Code-KR	Korean	Yes	1-3	
EUC_TW	Extended UNIX Code-TW	Traditional Chinese, Taiwanese	Yes	1-3	
GB18030	National Standard	Chinese	No	1-2	
GBK	Extended National Standard	Simplified Chinese	No	1-2	WIN936, Windows936
ISO_8859_5	ISO 8859-5, ECMA 113	Latin/Cyrillic	Yes	1	
ISO_8859_6	ISO 8859-6, ECMA 114	Latin/Arabic	Yes	1	
ISO_8859_7	ISO 8859-7, ECMA 118	Latin/Greek	Yes	1	
ISO_8859_8	ISO 8859-8, ECMA 121	Latin/Hebrew	Yes	1	
JOHAB	JOHA	Korean (Hangul)	Yes	1-3	
KOI8	KOI8-R(U)	Cyrillic	Yes	1	KOI8R
LATIN1	ISO 8859-1, ECMA 94	Western European	Yes	1	ISO88591

[1] **Not all APIs support all the listed character sets. For example, the JDBC driver does not support MULE_INTERNAL, LATIN6, LATIN8, and LATIN10.**

Name	Description	Language	Server?	Bytes/Char	Aliases
LATIN2	ISO 8859-2, ECMA 94	Central European	Yes	1	ISO88592
LATIN3	ISO 8859-3, ECMA 94	South European	Yes	1	ISO88593
LATIN4	ISO 8859-4, ECMA 94	North European	Yes	1	ISO88594
LATIN5	ISO 8859-9, ECMA 128	Turkish	Yes	1	ISO88599
LATIN6	ISO 8859-10, ECMA 144	Nordic	Yes	1	ISO885910
LATIN7	ISO 8859-13	Baltic	Yes	1	ISO885913
LATIN8	ISO 8859-14	Celtic	Yes	1	ISO885914
LATIN9	ISO 8859-15	LATIN1 with Euro and accents	Yes	1	ISO885915
LATIN10	ISO 8859-16, ASRO SR 14111	Romanian	Yes	1	ISO885916
MULE_ INTERNAL	Mule internal code	Multilingual Emacs	Yes	1-4	
SJIS	Shift JIS	Japanese	No	1-2	Mskanji, ShiftJIS, WIN932, Windows932
SQL_ASCII	unspecified[2]	any	No	1	
UHC	Unified Hangul Code	Korean	No	1-2	WIN949, Windows949
UTF8	Unicode, 8-bit	all	Yes	1-4	Unicode
WIN866	Windows CP866	Cyrillic	Yes	1	ALT
WIN874	Windows CP874	Thai	Yes	1	
WIN1250	Windows CP1250	Central European	Yes	1	
WIN1251	Windows CP1251	Cyrillic	Yes	1	WIN
WIN1252	Windows CP1252	Western European	Yes	1	

[2] The SQL_ASCII setting behaves considerably differently from the other settings. Byte values 0-127 are interpreted according to the ASCII standard, while byte values 128-255 are taken as uninterpreted characters. If you are working with any non-ASCII data, it is unwise to use the SQL_ASCII setting as a client encoding. SQL_ASCII is not supported as a server encoding.

Name	Description	Language	Server?	Bytes/Char	Aliases
WIN1253	Windows CP1253	Greek	Yes	1	
WIN1254	Windows CP1254	Turkish	Yes	1	
WIN1255	Windows CP1255	Hebrew	Yes	1	
WIN1256	Windows CP1256	Arabic	Yes	1	
WIN1257	Windows CP1257	Baltic	Yes	1	
WIN1258	Windows CP1258	Vietnamese	Yes	1	ABC, TCVN, TCVN5712, VSCII

Setting the Character Set

gpinitsystem defines the default character set for a Greenplum Database system by reading the setting of the `ENCODING` parameter in the `gp_init_config` file at initialization time. The default character set is `UNICODE` or `UTF8`.

You can create a database with a different character set besides what is used as the system-wide default. For example:

```
=> CREATE DATABASE korean WITH ENCODING 'EUC_KR';
```

> **Important:** Although you can specify any encoding you want for a database, it is unwise to choose an encoding that is not what is expected by the locale you have selected. The `LC_COLLATE` and `LC_CTYPE` settings imply a particular encoding, and locale-dependent operations (such as sorting) are likely to misinterpret data that is in an incompatible encoding.

Since these locale settings are frozen by gpinitsystem, the apparent flexibility to use different encodings in different databases is more theoretical than real.

One way to use multiple encodings safely is to set the locale to `C` or `POSIX` during initialization time, thus disabling any real locale awareness.

Character Set Conversion Between Server and Client

Greenplum Database supports automatic character set conversion between server and client for certain character set combinations. The conversion information is stored in the master `pg_conversion` system catalog table. Greenplum Database comes with some predefined conversions or you can create a new conversion using the SQL command `CREATE CONVERSION`.

Table 8: Client/Server Character Set Conversions

Server Character Set	Available Client Character Sets
BIG5	not supported as a server encoding
EUC_CN	EUC_CN, MULE_INTERNAL, UTF8
EUC_JP	EUC_JP, MULE_INTERNAL, SJIS, UTF8
EUC_KR	EUC_KR, MULE_INTERNAL, UTF8
EUC_TW	EUC_TW, BIG5, MULE_INTERNAL, UTF8
GB18030	not supported as a server encoding
GBK	not supported as a server encoding
ISO_8859_5	ISO_8859_5, KOI8, MULE_INTERNAL, UTF8, WIN866, WIN1251
ISO_8859_6	ISO_8859_6, UTF8
ISO_8859_7	ISO_8859_7, UTF8
ISO_8859_8	ISO_8859_8, UTF8
JOHAB	JOHAB, UTF8
KOI8	KOI8, ISO_8859_5, MULE_INTERNAL, UTF8, WIN866, WIN1251
LATIN1	LATIN1, MULE_INTERNAL, UTF8
LATIN2	LATIN2, MULE_INTERNAL, UTF8, WIN1250
LATIN3	LATIN3, MULE_INTERNAL, UTF8
LATIN4	LATIN4, MULE_INTERNAL, UTF8
LATIN5	LATIN5, UTF8
LATIN6	LATIN6, UTF8
LATIN7	LATIN7, UTF8
LATIN8	LATIN8, UTF8
LATIN9	LATIN9, UTF8
LATIN10	LATIN10, UTF8
MULE_INTERNAL	MULE_INTERNAL, BIG5, EUC_CN, EUC_JP, EUC_KR, EUC_TW, ISO_8859_5, KOI8, LATIN1 to LATIN4, SJIS, WIN866, WIN1250, WIN1251
SJIS	not supported as a server encoding

Server Character Set	Available Client Character Sets
SQL_ASCII	not supported as a server encoding
UHC	not supported as a server encoding
UTF8	all supported encodings
WIN866	WIN866
ISO_8859_5	KOI8, MULE_INTERNAL, UTF8, WIN1251
WIN874	WIN874, UTF8
WIN1250	WIN1250, LATIN2, MULE_INTERNAL, UTF8
WIN1251	WIN1251, ISO_8859_5, KOI8, MULE_INTERNAL, UTF8, WIN866
WIN1252	WIN1252, UTF8
WIN1253	WIN1253, UTF8
WIN1254	WIN1254, UTF8
WIN1255	WIN1255, UTF8
WIN1256	WIN1256, UTF8
WIN1257	WIN1257, UTF8
WIN1258	WIN1258, UTF8

To enable automatic character set conversion, you have to tell Greenplum Database the character set (encoding) you would like to use in the client. There are several ways to accomplish this:

- Using the `\encoding` command in `psql`, which allows you to change client encoding on the fly.
- Using `SET client_encoding TO`. Setting the client encoding can be done with this SQL command:

  ```
  => SET CLIENT_ENCODING TO 'value';
  ```

  To query the current client encoding:

  ```
  => SHOW client_encoding;
  ```

  To return to the default encoding:

  ```
  => RESET client_encoding;
  ```

- Using the `PGCLIENTENCODING` environment variable. When `PGCLIENTENCODING` is defined in the client's environment, that client encoding is automatically selected when a connection to the server is made. (This can subsequently be overridden using any of the other methods mentioned above.)
- Setting the configuration parameter `client_encoding`. If `client_encoding` is set in the master `postgresql.conf` file, that client encoding is automatically selected when a connection to Greenplum Database is made. (This can subsequently be overridden using any of the other methods mentioned above.)

If the conversion of a particular character is not possible — suppose you chose `EUC_JP` for the server and `LATIN1` for the client, then some Japanese characters do not have a representation in `LATIN1` — then an error is reported.

If the client character set is defined as `SQL_ASCII`, encoding conversion is disabled, regardless of the server's character set. The use of `SQL_ASCII` is unwise unless you are working with all-ASCII data. `SQL_ASCII` is not supported as a server encoding.

Chapter 8

Initializing a Greenplum Database System

This chapter describes how to initialize a Greenplum Database database system. The instructions in this chapter assume you have already installed the Greenplum Database software on all of the hosts in the system according to the instructions in *Configuring Your Systems and Installing Greenplum*.

This chapter contains the following topics:

Overview

Because Greenplum Database is distributed, the process for initializing a Greenplum Database management system (DBMS) involves initializing several individual PostgreSQL database instances (called *segment instances* in Greenplum).

Each database instance (the master and all segments) must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS. Greenplum provides its own version of `initdb` called *`gpinitsystem`*, which takes care of initializing the database on the master and on each segment instance, and starting each instance in the correct order.

After the Greenplum Database database system has been initialized and started, you can then create and manage databases as you would in a regular PostgreSQL DBMS by connecting to the Greenplum master.

Initializing Greenplum Database

These are the high-level tasks for initializing Greenplum Database:

1. Make sure you have completed all of the installation tasks described in *Configuring Your Systems and Installing Greenplum*.
2. Create a host file that contains the host addresses of your *segments*. See *Creating the Initialization Host File*.
3. Create your Greenplum Database system configuration file. See *Creating the Greenplum Database Configuration File*.
4. By default, Greenplum Database will be initialized using the locale of the master host system. Make sure this is the correct locale you want to use, as some locale options cannot be changed after initialization. See *Configuring Localization Settings* for more information.
5. Run the Greenplum Database initialization utility on the master host. See *Running the Initialization Utility*.

Creating the Initialization Host File

The *gpinitsystem* utility requires a host file that contains the list of addresses for each segment host. The initialization utility determines the number of segment instances per host by the number of host addresses listed per host times the number of data directory locations specified in the `gpinitsystem_config` file.

This file should only contain *segment* host addresses (not the master or standby master). For segment machines with more than one network interface, this file should list the host address names for each interface — one per line.

> **Note:** The Greenplum Database segment host naming convention is *sdwN* where *sdw* is a prefix and *N* is an integer. For example, on a Greenplum Database DCA system, segment host names would be `sdw1`, `sdw2` and so on. If hosts have multiple unbonded NICs, the convention is to append a dash (`-`) and number to the host name. For example, `sdw1-1` and `sdw1-2` are the two interface names for host `sdw1`. However, NIC bonding is recommended to create a load-balanced, fault-tolerant network.

To create the initialization host file

1. Log in as `gpadmin`.

```
$ su - gpadmin
```

2. Create a file named `hostfile_gpinitsystem`. In this file add the host address name(s) of your *segment* host interfaces, one name per line, no extra lines or spaces. For example, if you have four segment hosts with two network interfaces each:

```
sdw1-1
sdw1-2
sdw2-1
sdw2-2
sdw3-1
sdw3-2
sdw4-1
sdw4-2
```

3. Save and close the file.

> **Note:** If you are not sure of the host names and/or interface address names used by your machines, look in the `/etc/hosts` file.

Creating the Greenplum Database Configuration File

Your Greenplum Database configuration file tells the *gpinitsystem* utility how you want to configure your Greenplum Database system. An example configuration file can be found in `$GPHOME/docs/cli_help/gpconfigs/gpinitsystem_config`.

To create a gpinitsystem_config file

1. Log in as `gpadmin`.

```
$ su - gpadmin
```

2. Make a copy of the `gpinitsystem_config` file to use as a starting point. For example:

```
$ cp $GPHOME/docs/cli_help/gpconfigs/gpinitsystem_config \
     /home/gpadmin/gpconfigs/gpinitsystem_config
```

3. Open the file you just copied in a text editor.

   Set all of the required parameters according to your environment. See *gpinitsystem* for more information. A Greenplum Database system must contain a master instance and at *least two* segment instances (even if setting up a single node system).

   The `DATA_DIRECTORY` parameter is what determines how many segments per host will be created. If your segment hosts have multiple network interfaces, and you used their interface address names in your host file, the number of segments will be evenly spread over the number of available interfaces.

   Here is an example of the *required* parameters in the `gpinitsystem_config` file:

```
ARRAY_NAME="EMC Greenplum DW"
SEG_PREFIX=gpseg
PORT_BASE=40000
declare -a DATA_DIRECTORY=(/data1/primary /data1/primary
/data1/primary /data2/primary /data2/primary /data2/primary)
MASTER_HOSTNAME=mdw
MASTER_DIRECTORY=/data/master
MASTER_PORT=5432
TRUSTED_SHELL=ssh
CHECK_POINT_SEGMENTS=8
ENCODING=UNICODE
```

4. (Optional) If you want to deploy mirror segments, uncomment and set the mirroring parameters according to your environment. Here is an example of the *optional* mirror parameters in the `gpinitsystem_config` file:

```
MIRROR_PORT_BASE=7000
REPLICATION_PORT_BASE=8000
MIRROR_REPLICATION_PORT_BASE=9000
declare -a MIRROR_DATA_DIRECTORY=(/data1/mirror /data1/mirror /data1/mirror /
data2/mirror /data2/mirror /data2/mirror)
```

   Note: You can initialize your Greenplum system with primary segments only and deploy mirrors later using the *gpaddmirrors* utility.

5. Save and close the file.

Running the Initialization Utility

The *gpinitsystem* utility will create a Greenplum Database system using the values defined in the configuration file.

To run the initialization utility

1. Run the following command referencing the path and file name of your initialization configuration file (`gpinitsystem_config`) and host file (`hostfile_gpinitsystem`). For example:

   ```
   $ cd ~
   $ gpinitsystem -c gpconfigs/gpinitsystem_config -h gpconfigs/hostfile_gpinitsystem
   ```

   For a fully redundant system (with a standby master and a *spread* mirror configuration) include the `-s` and `-S` options. For example:

   ```
   $ gpinitsystem -c gpconfigs/gpinitsystem_config -h gpconfigs/hostfile_gpinitsystem
    \
     -s standby_master_hostname -S
   ```

2. The utility will verify your setup information and make sure it can connect to each host and access the data directories specified in your configuration. If all of the pre-checks are successful, the utility will prompt you to confirm your configuration. For example:

   ```
   => Continue with Greenplum creation? Yy/Nn
   ```

3. Press `y` to start the initialization.
4. The utility will then begin setup and initialization of the master instance and each segment instance in the system. Each segment instance is set up in parallel. Depending on the number of segments, this process can take a while.
5. At the end of a successful setup, the utility will start your Greenplum Database system. You should see:

   ```
   => Greenplum Database instance successfully created.
   ```

Troubleshooting Initialization Problems

If the utility encounters any errors while setting up an instance, the entire process will fail, and could possibly leave you with a partially created system. Refer to the error messages and logs to determine the cause of the failure and where in the process the failure occurred. Log files are created in `~/gpAdminLogs`.

Depending on when the error occurred in the process, you may need to clean up and then try the gpinitsystem utility again. For example, if some segment instances were created and some failed, you may need to stop `postgres` processes and remove any utility-created data directories from your data storage area(s). A backout script is created to help with this cleanup if necessary.

Using the Backout Script

If the gpinitsystem utility fails, it will create the following backout script if it has left your system in a partially installed state:

`~/gpAdminLogs/backout_gpinitsystem_<user>_<timestamp>`

You can use this script to clean up a partially created Greenplum Database system. This backout script will remove any utility-created data directories, `postgres` processes, and log files. After correcting the error that caused gpinitsystem to fail and running the backout script, you should be ready to retry initializing your Greenplum Database array.

The following example shows how to run the backout script:

```
$ sh backout_gpinitsystem_gpadmin_20071031_121053
```

Setting Greenplum Environment Variables

You must configure your environment on the Greenplum Database master (and standby master). A `greenplum_path.sh` file is provided in your `$GPHOME` directory with environment variable settings for Greenplum Database. You can source this file in the `gpadmin` user's startup shell profile (such as `.bashrc`).

The Greenplum Database management utilities also require that the `MASTER_DATA_DIRECTORY` environment variable be set. This should point to the directory created by the gpinitsystem utility in the master data directory location.

> **Note:** The `greenplum_path.sh` script changes the operating environment in order to support running the Greenplum Database-specific utilities. These same changes to the environment can negatively affect the operation of other system-level utilities, such as `ps` or `yum`. Use separate accounts for performing system administration and database administration, instead of attempting to perform both functions as `gpadmin`.

To set up your user environment for Greenplum

1. Make sure you are logged in as `gpadmin`:

```
$ su - gpadmin
```

2. Open your profile file (such as `.bashrc`) in a text editor. For example:

```
$ vi ~/.bashrc
```

3. Add lines to this file to source the `greenplum_path.sh` file and set the `MASTER_DATA_DIRECTORY` environment variable. For example:

```
source /usr/local/greenplum-db/greenplum_path.sh
export MASTER_DATA_DIRECTORY=/data/master/gpseg-1
```

4. (Optional) You may also want to set some client session environment variables such as `PGPORT`, `PGUSER` and `PGDATABASE` for convenience. For example:

```
export PGPORT=5432
export PGUSER=gpadmin export PGDATABASE=default_login_database_name
```

5. (Optional) If you use RHEL 7 or CentOS 7, add the following line to the end of the `.bashrc` file to enable using the `ps` command in the `greenplum_path.sh` environment:

```
export LD_PRELOAD=/lib64/libz.so.1 ps
```

6. Save and close the file.
7. After editing the profile file, source it to make the changes active. For example:

```
$ source ~/.bashrc
```

8. If you have a standby master host, copy your environment file to the standby master as well. For example:

```
$ cd ~
$ scp .bashrc standby_hostname:`pwd`
```

> **Note:** The `.bashrc` file should not produce any output. If you wish to have a message display to users upon logging in, use the `.profile` file instead.

Next Steps

After your system is up and running, the next steps are:

- *Securing the Greenplum Database System*
- *Creating Databases and Loading Data*

Securing the Greenplum Database System

When a Greenplum Database system is initialized, a host-based authentication configuration file (`pg_hba.conf`) is created that allows only local connections by the `gpadmin` role (or whatever system user ran `gpinitsystem`). If you would like other users or client machines to be able to connect to Greenplum Database, you must give them access by adding entries to the `pg_hba.conf` file.

> **Warning:** The authentication method for local database connections is initially `trust`, which means that Greenplum Database trusts any user logged in to the master server to access the database system using whatever database role they specify. Trust authentication is insufficiently secure for a production database system. You should replace all instances of `trust` authentication in the `pg_hba.conf` file with a stronger authentication method, such as `ident` for local connections or `md5` passwords for remote connections.

See "Configuring Client Authentication" in the *Greenplum Database Security Configuration Guide* for more about configuring authentication methods. See "Managing Roles and Privileges" in the *Greenplum Database Administrator Guide* for help creating database roles and managing users' access privileges in databases.

Creating Databases and Loading Data

After verifying your installation, you may want to begin creating databases and loading data. See *Defining Database Objects* and *Loading and Unloading Data* in the *Greenplum Database Administrator Guide* for more information about creating databases, schemas, tables, and other database objects in Greenplum Database and loading your data.

Chapter 9

Installation Management Utilities

This appendix provides references for the command-line management utilities used to install and initialize a Greenplum Database system. For a full reference of all Greenplum Database utilities, see the *Greenplum Database Utility Guide*.

The following Greenplum Database management utilities are located in `$GPHOME/bin`.

gpactivatestandby

Activates a standby master host and makes it the active master for the Greenplum Database system.

Synopsis

```
gpactivatestandby -d standby_master_datadir [-f] [-a] [-q]
    [-l logfile_directory]

gpactivatestandby -v

gpactivatestandby -? | -h | --help
```

Description

The `gpactivatestandby` utility activates a backup, standby master host and brings it into operation as the active master instance for a Greenplum Database system. The activated standby master effectively becomes the Greenplum Database master, accepting client connections on the master port.

When you initialize a standby master, the default is to use the same port as the active master. For information about the master port for the standby master, see *gpinitstandby*.

You must run this utility from the master host you are activating, not the failed master host you are disabling. Running this utility assumes you have a standby master host configured for the system (see *gpinitstandby*).

The utility will perform the following steps:

- Stops the synchronization process (`walreceiver`) on the standby master
- Updates the system catalog tables of the standby master using the logs
- Activates the standby master to be the new active master for the system
- Restarts the Greenplum Database system with the new master host

A backup, standby Greenplum master host serves as a 'warm standby' in the event of the primary Greenplum master host becoming non-operational. The standby master is kept up to date by transaction log replication processes (the `walsender` and `walreceiver`), which run on the primary master and standby master hosts and keep the data between the primary and standby master hosts synchronized.

If the primary master fails, the log replication process is shutdown, and the standby master can be activated in its place by using the `gpactivatestandby` utility. Upon activation of the standby master, the replicated logs are used to reconstruct the state of the Greenplum master host at the time of the last successfully committed transaction.

In order to use `gpactivatestandby` to activate a new primary master host, the master host that was previously serving as the primary master cannot be running. The utility checks for a `postmaster.pid` file in the data directory of the disabled master host, and if it finds it there, it will assume the old master host is still active. In some cases, you may need to remove the `postmaster.pid` file from the disabled master host data directory before running `gpactivatestandby` (for example, if the disabled master host process was terminated unexpectedly).

After activating a standby master, run `ANALYZE` to update the database query statistics. For example:

```
psql dbname -c 'ANALYZE;'
```

After you activate the standby master as the primary master, the Greenplum Database system no longer has a standby master configured. You might want to specify another host to be the new standby with the *gpinitstandby* utility.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-d *standby_master_datadir*

The absolute path of the data directory for the master host you are activating.

If this option is not specified, `gpactivatestandby` uses the value specified by the environment variable `MASTER_DATA_DIRECTORY` of the master host you are activating.

If a directory cannot be determined, the utility returns an error.

-f (force activation)

Use this option to force activation of the backup master host. Use this option only if you are sure that the standby and primary master hosts are consistent.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-v (show utility version)

Displays the version, status, last updated date, and check sum of this utility.

-? | -h | --help (help)

Displays the online help.

Example

Activate the standby master host and make it the active master instance for a Greenplum Database system (run from backup master host you are activating):

```
gpactivatestandby -d /gpdata
```

See Also

gpinitsystem, *gpinitstandby*

gpaddmirrors

Adds mirror segments to a Greenplum Database system that was initially configured without mirroring.

Synopsis

```
gpaddmirrors [-p port_offset] [-m datadir_config_file [-a]] [-s]
   [-d master_data_directory] [-B parallel_processes] [-l logfile_directory]
   [-v]

gpaddmirrors -i mirror_config_file [-a] [-d master_data_directory]
   [-B parallel_processes] [-l logfile_directory] [-v]

gpaddmirrors -o output_sample_mirror_config [-s] [-m datadir_config_file]

gpaddmirrors -?

gpaddmirrors --version
```

Description

The `gpaddmirrors` utility configures mirror segment instances for an existing Greenplum Database system that was initially configured with primary segment instances only. The utility will create the mirror instances and begin the online replication process between the primary and mirror segment instances. Once all mirrors are synchronized with their primaries, your Greenplum Database system is fully data redundant.

> **Important:** During the online replication process, Greenplum Database should be in a quiescent state, workloads and other queries should not be running.

By default, the utility will prompt you for the file system location(s) where it will create the mirror segment data directories. If you do not want to be prompted, you can pass in a file containing the file system locations using the `-m` option.

The mirror locations and ports must be different than your primary segment data locations and ports. If you have created additional filespaces, you will also be prompted for mirror locations for each of your filespaces.

The utility creates a unique data directory for each mirror segment instance in the specified location using the predefined naming convention. There must be the same number of file system locations declared for mirror segment instances as for primary segment instances. It is OK to specify the same directory name multiple times if you want your mirror data directories created in the same location, or you can enter a different data location for each mirror. Enter the absolute path. For example:

```
Enter mirror segment data directory location 1 of 2 > /gpdb/mirror
Enter mirror segment data directory location 2 of 2 > /gpdb/mirror
```

OR

```
Enter mirror segment data directory location 1 of 2 > /gpdb/m1
Enter mirror segment data directory location 2 of 2 > /gpdb/m2
```

Alternatively, you can run the `gpaddmirrors` utility and supply a detailed configuration file using the `-i` option. This is useful if you want your mirror segments on a completely different set of hosts than your primary segments. The format of the mirror configuration file is:

```
filespaceOrder=[filespace1_fsname[:filespace2_fsname:...]
mirror[content]=content:address:port:mir_replication_port:pri_replication_
port:fselocation[:fselocation:...]
```

For example (if you do not have additional filespaces configured besides the default *pg_system* filespace):

```
filespaceOrder=
mirror0=0:sdw1-1:60000:61000:62000:/gpdata/mir1/gp0
mirror1=1:sdw1-1:60001:61001:62001:/gpdata/mir2/gp1
```

The `gp_segment_configuration`, `pg_filespace`, and `pg_filespace_entry` system catalog tables can help you determine your current primary segment configuration so that you can plan your mirror segment configuration. For example, run the following query:

```
=# SELECT dbid, content, address as host_address, port,
   replication_port, fselocation as datadir
   FROM gp_segment_configuration, pg_filespace_entry
   WHERE dbid=fsedbid
   ORDER BY dbid;
```

If creating your mirrors on alternate mirror hosts, the new mirror segment hosts must be pre-installed with the Greenplum Database software and configured exactly the same as the existing primary segment hosts.

You must make sure that the user who runs `gpaddmirrors` (the `gpadmin` user) has permissions to write to the data directory locations specified. You may want to create these directories on the segment hosts and `chown` them to the appropriate user before running `gpaddmirrors`.

> **Note:** This utility uses secure shell (SSH) connections between systems to perform its tasks. In large Greenplum Database deployments, cloud deployments, or deployments with a large number of segments per host, this utility may exceed the host's maximum threshold for unauthenticated connections. Consider updating the SSH `MaxStartups` configuration parameter to increase this threshold. For more information about SSH configuration options, refer to the SSH documentation for your Linux distribution.

Options

-a (do not prompt)

Run in quiet mode - do not prompt for information. Must supply a configuration file with either `-m` or `-i` if this option is used.

-B *parallel_processes*

The number of mirror setup processes to start in parallel. If not specified, the utility will start up to 10 parallel processes depending on how many mirror segment instances it needs to set up.

-d *master_data_directory*

The master data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

-i *mirror_config_file*

A configuration file containing one line for each mirror segment you want to create. You must have one mirror segment listed for each primary segment in the system. The format of this file is as follows (as per attributes in the `gp_segment_configuration`, `pg_filespace`, and `pg_filespace_entry` catalog tables):

```
filespaceOrder=[filespace1_fsname[:filespace2_fsname:...]
mirror[content]=content:address:port:mir_replication_port:pri_replication_
port:fselocation[:fselocation:...]
```

Note that you only need to specify a name for `filespaceOrder` if your system has multiple filespaces configured. If your system does not have additional filespaces configured besides the default `pg_system` filespace, this file will only have one location (for the default data directory filespace, `pg_system`). `pg_system` does not need to be listed in the `filespaceOrder` line. It will always be the first *fselocation* listed after *replication_port*.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-m *datadir_config_file*

A configuration file containing a list of file system locations where the mirror data directories will be created. If not supplied, the utility prompts you for locations. Each line in the file specifies a mirror data directory location. For example:

```
/gpdata/m1
/gpdata/m2
/gpdata/m3
/gpdata/m4
```

If your system has additional filespaces configured in addition to the default `pg_system` filespace, you must also list file system locations for each filespace as follows:

```
filespace filespace1
/gpfs1/m1
/gpfs1/m2
/gpfs1/m3
/gpfs1/m4
```

-o *output_sample_mirror_config*

If you are not sure how to lay out the mirror configuration file used by the `-i` option, you can run `gpaddmirrors` with this option to generate a sample mirror configuration file based on your primary segment configuration. The utility will prompt you for your mirror segment data directory locations (unless you provide these in a file using `-m`). You can then edit this file to change the host names to alternate mirror hosts if necessary.

-p *port_offset*

Optional. This number is used to calculate the database ports and replication ports used for mirror segments. The default offset is 1000. Mirror port assignments are calculated as follows:

primary port + offset = mirror database port

primary port + (2 * offset) = mirror replication port

primary port + (3 * offset) = primary replication port

For example, if a primary segment has port 50001, then its mirror will use a database port of 51001, a mirror replication port of 52001, and a primary replication port of 53001 by default.

-s (spread mirrors)

Spreads the mirror segments across the available hosts. The default is to group a set of mirror segments together on an alternate host from their primary segment set. Mirror spreading will place each mirror on a different host within the Greenplum Database array. Spreading is only allowed if there is a sufficient number of hosts in the array (number of hosts is greater than the number of segment instances per host).

-v (verbose)

Sets logging output to verbose.

--version (show utility version)

Displays the version of this utility.

-? (help)

Displays the online help.

Examples

Add mirroring to an existing Greenplum Database system using the same set of hosts as your primary data. Calculate the mirror database and replication ports by adding 100 to the current primary segment port numbers:

```
$ gpaddmirrors -p 100
```

Add mirroring to an existing Greenplum Database system using a different set of hosts from your primary data:

```
$ gpaddmirrors -i mirror_config_file
```

Where `mirror_config_file` looks something like this (if you do not have additional filespaces configured besides the default `pg_system` filespace):

```
filespaceOrder=
mirror0=0:sdw1-1:52001:53001:54001:/gpdata/mir1/gp0
mirror1=1:sdw1-2:52002:53002:54002:/gpdata/mir2/gp1
mirror2=2:sdw2-1:52001:53001:54001:/gpdata/mir1/gp2
mirror3=3:sdw2-2:52002:53002:54002:/gpdata/mir2/gp3
```

Output a sample mirror configuration file to use with `gpaddmirrors -i`:

```
$ gpaddmirrors -o /home/gpadmin/sample_mirror_config
```

See Also

gpinitsystem, *gpinitstandby*, *gpactivatestandby*

gpcheck

Verifies and validates Greenplum Database platform settings.

Synopsis

```
gpcheck {{-f | --file} hostfile_gpcheck | {-h | --host} host_ID| --local }
   [-m master_host] [-s standby_master_host] [--stdout | --zipout]
   [--config config_file]

gpcheck --zipin gpcheck_zipfile

gpcheck -?

gpcheck --version
```

Description

The `gpcheck` utility determines the platform on which you are running Greenplum Database and validates various platform-specific configuration settings. `gpcheck` can use a host file or a file previously created with the `--zipout` option to validate platform settings. At the end of a successful validation process, `GPCHECK_NORMAL` message displays. If `GPCHECK_ERROR` displays, one or more validation checks failed. You can use also `gpcheck` to gather and view platform settings on hosts without running validation checks.

You should run `gpcheck` as `root`. If you do not run `gpcheck` as `root`, the utility displays a warning message and will not be able to validate all configuration settings; Only some of these settings will be validated.

Options

--config *config_file*

The name of a configuration file to use instead of the default file `$GPHOME/etc/gpcheck.cnf` (or `~/gpconfigs/gpcheck_dca_config` on the Dell EMC Greenplum Data Computing Appliance). This file specifies the OS-specific checks to run.

{-f | --file} *hostfile_gpcheck*

The name of a file that contains a list of hosts that `gpcheck` uses to validate platform-specific settings. This file should contain a single host name for all hosts in your Greenplum Database system (master, standby master, and segments). `gpcheck` uses SSH to connect to the hosts.

{--h | --host} *host_ID*

Checks platform-specific settings on the host in your Greenplum Database system specified by *host_ID*. `gpcheck` uses SSH to connect to the host.

--local

Checks platform-specific settings on the segment host where `gpcheck` is run. This option does not require SSH authentication.

-m *master_host*

This option is deprecated and will be removed in a future release.

-s *standby_master_host*

This option is deprecated and will be removed in a future release.

--stdout

Display collected host information from gpcheck. No checks or validations are performed.

--zipout

Save all collected data to a `.zip` file in the current working directory. `gpcheck` automatically creates the `.zip` file and names it gpcheck_timestamp.tar.gz. No checks or validations are performed.

--zipin *gpcheck_zipfile*

Use this option to decompress and check a `.zip` file created with the `--zipout` option. `gpcheck` performs validation tasks against the file you specify in this option.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Verify and validate the Greenplum Database platform settings by entering a host file:

```
# gpcheck -f hostfile_gpcheck
```

Save Greenplum Database platform settings to a zip file:

```
# gpcheck -f hostfile_gpcheck --zipout
```

Verify and validate the Greenplum Database platform settings using a zip file created with the `--zipout` option:

```
# gpcheck --zipin gpcheck_timestamp.tar.gz
```

View collected Greenplum Database platform settings:

```
# gpcheck -f hostfile_gpcheck --stdout
```

See Also

gpssh, *gpscp*, *gpcheckperf*

gpcheckperf

Verifies the baseline hardware performance of the specified hosts.

Synopsis

```
gpcheckperf -d test_directory [-d test_directory ...]
    {-f hostfile_gpcheckperf | - h hostname [-h hostname ...]}
    [-r ds] [-B block_size] [-S file_size] [-D] [-v|-V]

gpcheckperf -d temp_directory
    {-f hostfile_gpchecknet | - h hostname [-h hostname ...]}
    [ -r n|N|M [--duration time] [--netperf] ] [-D] [-v | -V]

gpcheckperf -?

gpcheckperf --version
```

Description

The `gpcheckperf` utility starts a session on the specified hosts and runs the following performance tests:

- **Disk I/O Test (dd test)** — To test the sequential throughput performance of a logical disk or file system, the utility uses the **dd** command, which is a standard UNIX utility. It times how long it takes to write and read a large file to and from disk and calculates your disk I/O performance in megabytes (MB) per second. By default, the file size that is used for the test is calculated at two times the total random access memory (RAM) on the host. This ensures that the test is truly testing disk I/O and not using the memory cache.
- **Memory Bandwidth Test (stream)** — To test memory bandwidth, the utility uses the STREAM benchmark program to measure sustainable memory bandwidth (in MB/s). This tests that your system is not limited in performance by the memory bandwidth of the system in relation to the computational performance of the CPU. In applications where the data set is large (as in Greenplum Database), low memory bandwidth is a major performance issue. If memory bandwidth is significantly lower than the theoretical bandwidth of the CPU, then it can cause the CPU to spend significant amounts of time waiting for data to arrive from system memory.
- **Network Performance Test (gpnetbench*)** — To test network performance (and thereby the performance of the Greenplum Database interconnect), the utility runs a network benchmark program that transfers a 5 second stream of data from the current host to each remote host included in the test. The data is transferred in parallel to each remote host and the minimum, maximum, average and median network transfer rates are reported in megabytes (MB) per second. If the summary transfer rate is slower than expected (less than 100 MB/s), you can run the network test serially using the `-r n` option to obtain per-host results. To run a full-matrix bandwidth test, you can specify `-r M` which will cause every host to send and receive data from every other host specified. This test is best used to validate if the switch fabric can tolerate a full-matrix workload.

To specify the hosts to test, use the `-f` option to specify a file containing a list of host names, or use the `-h` option to name single host names on the command-line. If running the network performance test, all entries in the host file must be for network interfaces within the same subnet. If your segment hosts have multiple network interfaces configured on different subnets, run the network test once for each subnet.

You must also specify at least one test directory (with `-d`). The user who runs `gpcheckperf` must have write access to the specified test directories on all remote hosts. For the disk I/O test, the test directories should correspond to your segment data directories (primary and/or mirrors). For the memory bandwidth and network tests, a temporary directory is required for the test program files.

Before using `gpcheckperf`, you must have a trusted host setup between the hosts involved in the performance test. You can use the utility `gpssh-exkeys` to update the known host files and exchange

public keys between hosts if you have not done so already. Note that `gpcheckperf` calls to `gpssh` and `gpscp`, so these Greenplum utilities must also be in your `$PATH`.

Options

-B *block_size*

Specifies the block size (in KB or MB) to use for disk I/O test. The default is 32KB, which is the same as the Greenplum Database page size. The maximum block size is 1 MB.

-d *test_directory*

For the disk I/O test, specifies the file system directory locations to test. You must have write access to the test directory on all hosts involved in the performance test. You can use the `-d` option multiple times to specify multiple test directories (for example, to test disk I/O of your primary and mirror data directories).

-d *temp_directory*

For the network and stream tests, specifies a single directory where the test program files will be copied for the duration of the test. You must have write access to this directory on all hosts involved in the test.

-D (display per-host results)

Reports performance results for each host for the disk I/O tests. The default is to report results for just the hosts with the minimum and maximum performance, as well as the total and average performance of all hosts.

--duration *time*

Specifies the duration of the network test in seconds (s), minutes (m), hours (h), or days (d). The default is 15 seconds.

-f *hostfile_gpcheckperf*

For the disk I/O and stream tests, specifies the name of a file that contains one host name per host that will participate in the performance test. The host name is required, and you can optionally specify an alternate user name and/or SSH port number per host. The syntax of the host file is one host per line as follows:

```
[username@]hostname[:ssh_port]
```

-f *hostfile_gpchecknet*

For the network performance test, all entries in the host file must be for host adresses within the same subnet. If your segment hosts have multiple network interfaces configured on different subnets, run the network test once for each subnet. For example (a host file containing segment host address names for interconnect subnet 1):

```
sdw1-1
sdw2-1
sdw3-1
```

-h *hostname*

Specifies a single host name (or host address) that will participate in the performance test. You can use the `-h` option multiple times to specify multiple host names.

--netperf

Specifies that the `netperf` binary should be used to perform the network test instead of the Greenplum network test. To use this option, you must download `netperf` from *http://www.netperf.org* and install it into `$GPHOME/bin/lib` on all Greenplum hosts (master and segments).

-r ds{n|N|M}

Specifies which performance tests to run. The default is `dsn`:

- Disk I/O test (`d`)
- Stream test (`s`)
- Network performance test in sequential (`n`), parallel (`N`), or full-matrix (`M`) mode. The optional `--duration` option specifies how long (in seconds) to run the network test. To use the parallel (`N`) mode, you must run the test on an *even* number of hosts.

  If you would rather use `netperf` (*http://www.netperf.org*) instead of the Greenplum network test, you can download it and install it into `$GPHOME/bin/lib` on all Greenplum hosts (master and segments). You would then specify the optional `--netperf` option to use the `netperf` binary instead of the default `gpnetbench*` utilities.

-S *file_size*

Specifies the total file size to be used for the disk I/O test for all directories specified with `-d`. *file_size* should equal two times total RAM on the host. If not specified, the default is calculated at two times the total RAM on the host where `gpcheckperf` is executed. This ensures that the test is truly testing disk I/O and not using the memory cache. You can specify sizing in KB, MB, or GB.

-v (verbose) | -V (very verbose)

Verbose mode shows progress and status messages of the performance tests as they are run. Very verbose mode shows all output messages generated by this utility.

--version

Displays the version of this utility.

-? (help)

Displays the online help.

Examples

Run the disk I/O and memory bandwidth tests on all the hosts in the file *host_file* using the test directory of */data1* and */data2*:

```
$ gpcheckperf -f hostfile_gpcheckperf -d /data1 -d /data2 -r ds
```

Run only the disk I/O test on the hosts named *sdw1* and *sdw2* using the test directory of */data1*. Show individual host results and run in verbose mode:

```
$ gpcheckperf -h sdw1 -h sdw2 -d /data1 -r d -D -v
```

Run the parallel network test using the test directory of */tmp,* where *hostfile_gpcheck_ic** specifies all network interface host address names within the same interconnect subnet:

```
$ gpcheckperf -f hostfile_gpchecknet_ic1 -r N -d /tmp
$ gpcheckperf -f hostfile_gpchecknet_ic2 -r N -d /tmp
```

Run the same test as above, but use `netperf` instead of the Greenplum network test (note that `netperf` must be installed in `$GPHOME/bin/lib` on all Greenplum hosts):

```
$ gpcheckperf -f hostfile_gpchecknet_ic1 -r N --netperf -d /tmp
$ gpcheckperf -f hostfile_gpchecknet_ic2 -r N --netperf -d /tmp
```

See Also

gpssh, *gpscp*, *gpcheck*

gpdeletesystem

Deletes a Greenplum Database system that was initialized using `gpinitsystem`.

Synopsis

```
gpdeletesystem -d master_data_directory [-B parallel_processes]
   [-f] [-l logfile_directory] [-D]

gpdeletesystem -?

gpdeletesystem -v
```

Description

The `gpdeletesystem` utility will perform the following actions:

- Stop all `postgres` processes (the segment instances and master instance).
- Deletes all data directories.

Before running `gpdeletesystem`:

- Move any backup files out of the master and segment data directories.
- Make sure that Greenplum Database is running.
- If you are currently in a segment data directory, change directory to another location. The utility fails with an error when run from within a segment data directory.

This utility will not uninstall the Greenplum Database software.

Options

-d *data_directory*

Required. The master host data directory.

-B *parallel_processes*

The number of segments to delete in parallel. If not specified, the utility will start up to 60 parallel processes depending on how many segment instances it needs to delete.

-f (force)

Force a delete even if backup files are found in the data directories. The default is to not delete Greenplum Database instances if backup files are present.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-D (debug)

Sets logging level to debug.

-? (help)

Displays the online help.

-v (show utility version)

Displays the version, status, last updated date, and check sum of this utility.

Examples

Delete a Greenplum Database system:

```
gpdeletesystem -d /gpdata/gp-1
```

Delete a Greenplum Database system even if backup files are present:

```
gpdeletesystem -d /gpdata/gp-1 -f
```

See Also

gpinitsystem

gpinitstandby

Adds and/or initializes a standby master host for a Greenplum Database system.

Synopsis

```
gpinitstandby { -s standby_hostname [-P port]
      [-F list_of_filespaces] | -r | -n }
      [-a] [-q] [-D] [-l logfile_directory]

gpinitstandby -v

gpinitstandby -?
```

Description

The `gpinitstandby` utility adds a backup, standby master host to your Greenplum Database system. If your system has an existing standby master host configured, use the `-r` option to remove it before adding the new standby master host.

Before running this utility, make sure that the Greenplum Database software is installed on the standby master host and that you have exchanged SSH keys between the hosts. It is recommended that the master port is set to the same port number on the master host and the backup master host.

This utility should be run on the currently active *primary* master host. See the *Greenplum Database Installation Guide* for instructions.

The utility performs the following steps:

- Updates the Greenplum Database system catalog to remove the existing standby master host information (if the `-r` option is supplied)
- Updates the Greenplum Database system catalog to add the new standby master host information
- Edits the `pg_hba.conf` file of the Greenplum Database master to allow access from the newly added standby master.
- Sets up the standby master instance on the alternate master host
- Starts the synchronization process

A backup, standby master host serves as a 'warm standby' in the event of the primary master host becoming non-operational. The standby master is kept up to date by transaction log replication processes (the `walsender` and `walreceiver`), which run on the primary master and standby master hosts and keep the data between the primary and standby master hosts synchronized. If the primary master fails, the log replication process is shut down, and the standby master can be activated in its place by using the `gpactivatestandby` utility. Upon activation of the standby master, the replicated logs are used to reconstruct the state of the master host at the time of the last successfully committed transaction.

The activated standby master effectively becomes the Greenplum Database master, accepting client connections on the master port and performing normal master operations such as SQL command processing and workload management.

> **Important:** If the `gpinitstandby` utility previously failed to initialize the standby master, you must delete the files in the standby master data directory before running `gpinitstandby` again. The standby master data directory is not cleaned up after an initialization failure because it contains log files that can help in determining the reason for the failure.
>
> If an initialization failure occurs, a summary report file is generated in the standby host directory `/tmp`. The report file lists the directories on standby host that require clean up.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-D (debug)

Sets logging level to debug.

-F *list_of_filespaces*

A list of filespace names and the associated locations. Each filespace name and its location is separated by a colon. If there is more than one file space name, each pair (name and location) is separated by a comma. For example:

```
filespace1_name:fs1_location,filespace2_name:fs2_location
```

If this option is not specified, `gpinitstandby` prompts the user for the filespace names and locations.

If the list is not formatted correctly or number of filespaces do not match the number of filespaces already created in the system, `gpinitstandby` returns an error.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-n (restart standby master)

Specify this option to start a Greenplum Database standby master that has been configured but has stopped for some reason.

-P *port*

This option specifies the port that is used by the Greenplum Database standby master. The default is the same port used by the active Greenplum Database master.

If the Greenplum Database standby master is on the same host as the active master, the ports must be different. If the ports are the same for the active and standby master and the host is the same, the utility returns an error.

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-r (remove standby master)

Removes the currently configured standby master host from your Greenplum Database system.

-s standby_hostname

The host name of the standby master host.

-v (show utility version)

Displays the version, status, last updated date, and check sum of this utility.

-? (help)

Displays the online help.

Examples

Add a standby master host to your Greenplum Database system and start the synchronization process:

```
gpinitstandby -s host09
```

Start an existing standby master host and synchronize the data with the current primary master host:

```
gpinitstandby -n
```

Note: Do not specify the -n and -s options in the same command.

Add a standby master host to your Greenplum Database system specifying a different port:

```
gpinitstandby -s myhost -P 2222
```

If you specify the same host name as the active Greenplum Database master, the installed Greenplum Database software that is used as a standby master must be in a separate location from the active Greenplum Database master. Also, filespace locations that are used by the standby master must be different than the filespace locations used by the active Greenplum Database master.

Remove the existing standby master from your Greenplum system configuration:

```
gpinitstandby -r
```

See Also

gpinitsystem, *gpaddmirrors*, *gpactivatestandby*

gpinitsystem

Initializes a Greenplum Database system using configuration parameters specified in the `gpinitsystem_config` file.

Synopsis

```
gpinitsystem -c cluster_configuration_file
             [-h hostfile_gpinitsystem]
             [-B parallel_processes]
             [-p postgresql_conf_param_file]
             [-s standby_master_host [-P standby_master_port]
               [-F standby_master_filespaces]]
             [-m number | --max_connections=number]
             [-b size | --shared_buffers=size]
             [-n locale | --locale=locale] [--lc-collate=locale]
             [--lc-ctype=locale] [--lc-messages=locale]
             [--lc-monetary=locale] [--lc-numeric=locale]
             [--lc-time=locale]
             [-e password | --su_password=password]
             [-S] [-a] [-q] [-l logfile_directory] [-D]
             [-I input_configuration_file]
             [-O output_configuration_file]

gpinitsystem -v

gpinitsystem -h
```

Description

The `gpinitsystem` utility creates a Greenplum Database instance or writes an input configuration file using the values defined in a cluster configuration file and any command-line options that you provide. See *Initialization Configuration File Format* for more information about the configuration file. Before running this utility, make sure that you have installed the Greenplum Database software on all the hosts in the array.

With the `-O output_configuration_file` option, `gpinitsystem` does not create a new database instance but instead writes all provided configuration information to the specified output file. This file uses the `QD_PRIMARY_ARRAY` and `PRIMARY_ARRAY` parameters to define each member using its hostname, port, data directory, segment prefix, segment ID, and content ID. Details of the array configuration can be modified as necessary to match values available in a Greenplum Database backup, or can simply be used to recreate the same cluster configuration at a later time. Configuration files that use `QD_PRIMARY_ARRAY` and `PRIMARY_ARRAY` must be passed into `gpinitsystem` using the `-I input_configuration_file` option. See *Initialization Configuration File Format* for more information.

In a Greenplum Database DBMS, each database instance (the master and all segments) must be initialized across all of the hosts in the system in such a way that they can all work together as a unified DBMS. The `gpinitsystem` utility takes care of initializing the Greenplum master and each segment instance, and configuring the system as a whole.

Before running `gpinitsystem`, you must set the `$GPHOME` environment variable to point to the location of your Greenplum Database installation on the master host and exchange SSH keys between all host addresses in the array using `gpssh-exkeys`.

This utility performs the following tasks:

- Verifies that the parameters in the configuration file are correct.
- Ensures that a connection can be established to each host address. If a host address cannot be reached, the utility will exit.
- Verifies the locale settings.

- Displays the configuration that will be used and prompts the user for confirmation.
- Initializes the master instance.
- Initializes the standby master instance (if specified).
- Initializes the primary segment instances.
- Initializes the mirror segment instances (if mirroring is configured).
- Configures the Greenplum Database system and checks for errors.
- Starts the Greenplum Database system.

Note: This utility uses secure shell (SSH) connections between systems to perform its tasks. In large Greenplum Database deployments, cloud deployments, or deployments with a large number of segments per host, this utility may exceed the host's maximum threshold for unauthenticated connections. Consider updating the SSH `MaxStartups` configuration parameter to increase this threshold. For more information about SSH configuration options, refer to the SSH documentation for your Linux distribution.

Options

-a

Do not prompt the user for confirmation.

-B *parallel_processes*

The number of segments to create in parallel. If not specified, the utility will start up to 4 parallel processes at a time.

-c *cluster_configuration_file*

Required. The full path and filename of the configuration file, which contains all of the defined parameters to configure and initialize a new Greenplum Database system. See *Initialization Configuration File Format* for a description of this file. You must provide either the `-c cluster_configuration_file` option or the `-I input_configuration_file` option to `gpinitsystem`.

-D

Sets log output level to debug.

-h *hostfile_gpinitsystem*

Optional. The full path and filename of a file that contains the host addresses of your segment hosts. If not specified on the command line, you can specify the host file using the MACHINE_LIST_FILE parameter in the *gpinitsystem_config* file.

-I *input_configuration_file*

The full path and filename of an input configuration file, which defines the Greenplum Database members and segments using the `QD_PRIMARY_ARRAY` and `PRIMARY_ARRAY` parameters. The input configuration file is typically created by using gpinitsystem with the `-O output_configuration_file` option. You must provide either the `-c cluster_configuration_file` option or the `-I input_configuration_file` option to `gpinitsystem`.

--locale=*locale* | -n *locale*

Sets the default locale used by Greenplum Database. If not specified, the `LC_ALL`, `LC_COLLATE`, or `LANG` environment variable of the master host determines the locale. If these are not set, the default locale is `C` (`POSIX`). A locale identifier consists of a language identifier and a region identifier, and optionally a character set encoding. For example, `sv_SE` is Swedish as spoken in Sweden, `en_US` is U.S. English, and `fr_CA` is French Canadian. If more than one character set can be useful for a locale, then the specifications look like this: `en_US.UTF-8` (locale specification and character set encoding). On most systems, the command `locale` will show the locale environment settings and `locale -a` will show a list of all available locales.

--lc-collate=*locale*

Similar to `--locale`, but sets the locale used for collation (sorting data). The sort order cannot be changed after Greenplum Database is initialized, so it is important to choose a collation locale that is compatible with the character set encodings that you plan to use for your data. There is a special collation name of `C` or `POSIX` (byte-order sorting as opposed to dictionary-order sorting). The `C` collation can be used with any character encoding.

--lc-ctype=*locale*

Similar to `--locale`, but sets the locale used for character classification (what character sequences are valid and how they are interpreted). This cannot be changed after Greenplum Database is initialized, so it is important to choose a character classification locale that is compatible with the data you plan to store in Greenplum Database.

--lc-messages=*locale*

Similar to `--locale`, but sets the locale used for messages output by Greenplum Database. The current version of Greenplum Database does not support multiple locales for output messages (all messages are in English), so changing this setting will not have any effect.

--lc-monetary=*locale*

Similar to `--locale`, but sets the locale used for formatting currency amounts.

--lc-numeric=*locale*

Similar to `--locale`, but sets the locale used for formatting numbers.

--lc-time=*locale*

Similar to `--locale`, but sets the locale used for formatting dates and times.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

--max_connections=number | -m *number*

Sets the maximum number of client connections allowed to the master. The default is 250.

-O *output_configuration_file*

When used with the `-O` option, `gpinitsystem` does not create a new Greenplum Database cluster but instead writes the supplied cluster configuration information to the specified *`output_configuration_file`*. This file defines Greenplum Database members and segments using the `QD_PRIMARY_ARRAY`, `PRIMARY_ARRAY`, and `MIRROR_ARRAY` parameters, and can be later used with `-I` *`input_configuration_file`* to initialize a new cluster.

-p *postgresql_conf_param_file*

Optional. The name of a file that contains `postgresql.conf` parameter settings that you want to set for Greenplum Database. These settings will be used when the individual master and segment instances are initialized. You can also set parameters after initialization using the `gpconfig` utility.

-q

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

--shared_buffers=*size* | -b *size*

Sets the amount of memory a Greenplum server instance uses for shared memory buffers. You can specify sizing in kilobytes (kB), megabytes (MB) or gigabytes (GB). The default is 125MB.

-s *standby_master_host*

Optional. If you wish to configure a backup master host, specify the host name using this option. The Greenplum Database software must already be installed and configured on this host.

-P *standby_master_port*

Optional. If you configure a standby master host, specify its port number using this option. The Greenplum Database software must already be installed and configured on this host.

-F *standby_master_filespaces*

Optional. If you configure a standby master host, specify a list of filespace names and the associated locations using this option. Each filespace name and its location is separated by a colon. If there is more than one file space name, each pair (name and location) is separated by a comma. For example:

```
filespace1_name:fs1_location,filespace2_name:fs2_location
```

If this option is not specified, `gpinitstandby` prompts the user for the filespace names and locations.

If the list is not formatted correctly or number of filespaces do not match the number of filespaces already created in the system, `gpinitstandby` returns an error.

--su_password=*superuser_password* | -e *superuser_password*

Use this option to specify the password to set for the Greenplum Database superuser account (such as `gpadmin`). If this option is not specified, the default password `gparray` is assigned to the superuser account. You can use the `ALTER ROLE` command to change the password at a later time.

Recommended security best practices:

- Do not use the default password option for production environments.
- Change the password immediately after installation.

-S

If mirroring parameters are specified, spreads the mirror segments across the available hosts. The default is to group the set of mirror segments together on an alternate host from their primary segment set. Mirror spreading places each mirror on a different host within the Greenplum Database array. Spreading is only allowed if the number of hosts is greater than the number of segment instances.

-v

Displays the version of this utility.

-h

Displays the online help.

Initialization Configuration File Format

`gpinitsystem` requires a cluster configuration file with the following parameters defined. An example initialization configuration file can be found in `$GPHOME/docs/cli_help/gpconfigs/gpinitsystem_config`.

To avoid port conflicts between Greenplum Database and other applications, the Greenplum Database port numbers should not be in the range specified by the operating system parameter `net.ipv4.ip_local_port_range`. For example, if `net.ipv4.ip_local_port_range = 10000 65535`, you could set Greenplum Database base port numbers to these values.

```
PORT_BASE = 6000
MIRROR_PORT_BASE = 7000
REPLICATION_PORT_BASE = 8000
MIRROR_REPLICATION_PORT_BASE = 9000
```

ARRAY_NAME

Required. A name for the array you are configuring. You can use any name you like. Enclose the name in quotes if the name contains spaces.

MACHINE_LIST_FILE

Optional. Can be used in place of the `-h` option. This specifies the file that contains the list of segment host address names that comprise the Greenplum Database system. The master host is assumed to be the host from which you are running the utility and should not be included in this file. If your segment hosts have multiple network interfaces, then this file would include all addresses for the host. Give the absolute path to the file.

SEG_PREFIX

Required. This specifies a prefix that will be used to name the data directories on the master and segment instances. The naming convention for data directories in a Greenplum Database system is *SEG_PREFIXnumber* where *number* starts with 0 for segment instances (the master is always -1). So for example, if you choose the prefix `gpseg`, your master instance data directory would be named `gpseg-1`, and the segment instances would be named `gpseg0`, `gpseg1`, `gpseg2`, `gpseg3`, and so on.

PORT_BASE

Required. This specifies the base number by which primary segment port numbers are calculated. The first primary segment port on a host is set as `PORT_BASE`, and then incremented by one for each additional primary segment on that host. Valid values range from 1 through 65535.

DATA_DIRECTORY

Required. This specifies the data storage location(s) where the utility will create the primary segment data directories. The number of locations in the list dictate the number of primary segments that will get created per physical host (if multiple addresses for a host are listed in the host file, the number of segments will be spread evenly across the specified interface addresses). It is OK to list the same data storage area multiple times if you want your data directories created in the same location. The user who runs `gpinitsystem` (for example, the `gpadmin` user) must have permission to write to these directories. For example, this will create six primary segments per host:

```
declare -a DATA_DIRECTORY=(/data1/primary /data1/primary
/data1/primary /data2/primary /data2/primary /data2/primary)
```

MASTER_HOSTNAME

Required. The host name of the master instance. This host name must exactly match the configured host name of the machine (run the `hostname` command to determine the correct hostname).

MASTER_DIRECTORY

Required. This specifies the location where the data directory will be created on the master host. You must make sure that the user who runs `gpinitsystem` (for example, the `gpadmin` user) has permissions to write to this directory.

MASTER_PORT

Required. The port number for the master instance. This is the port number that users and client connections will use when accessing the Greenplum Database system.

TRUSTED_SHELL

Required. The shell the `gpinitsystem` utility uses to execute commands on remote hosts. Allowed values are `ssh`. You must set up your trusted host environment before running the `gpinitsystem` utility (you can use `gpssh-exkeys` to do this).

CHECK_POINT_SEGMENTS

Required. Maximum distance between automatic write ahead log (WAL) checkpoints, in log file segments (each segment is normally 16 megabytes). This will set the

`checkpoint_segments` parameter in the `postgresql.conf` file for each segment instance in the Greenplum Database system.

ENCODING

Required. The character set encoding to use. This character set must be compatible with the `--locale` settings used, especially `--lc-collate` and `--lc-ctype`. Greenplum Database supports the same character sets as PostgreSQL.

DATABASE_NAME

Optional. The name of a Greenplum Database database to create after the system is initialized. You can always create a database later using the `CREATE DATABASE` command or the `createdb` utility.

MIRROR_PORT_BASE

Optional. This specifies the base number by which mirror segment port numbers are calculated. The first mirror segment port on a host is set as `MIRROR_PORT_BASE`, and then incremented by one for each additional mirror segment on that host. Valid values range from 1 through 65535 and cannot conflict with the ports calculated by `PORT_BASE`.

REPLICATION_PORT_BASE

Optional. This specifies the base number by which the port numbers for the primary file replication process are calculated. The first replication port on a host is set as `REPLICATION_PORT_BASE`, and then incremented by one for each additional primary segment on that host. Valid values range from 1 through 65535 and cannot conflict with the ports calculated by `PORT_BASE` or `MIRROR_PORT_BASE`.

MIRROR_REPLICATION_PORT_BASE

Optional. This specifies the base number by which the port numbers for the mirror file replication process are calculated. The first mirror replication port on a host is set as `MIRROR_REPLICATION_PORT_BASE`, and then incremented by one for each additional mirror segment on that host. Valid values range from 1 through 65535 and cannot conflict with the ports calculated by `PORT_BASE`, `MIRROR_PORT_BASE`, or `REPLICATION_PORT_BASE`.

MIRROR_DATA_DIRECTORY

Optional. This specifies the data storage location(s) where the utility will create the mirror segment data directories. There must be the same number of data directories declared for mirror segment instances as for primary segment instances (see the `DATA_DIRECTORY` parameter). The user who runs `gpinitsystem` (for example, the `gpadmin` user) must have permission to write to these directories. For example:

```
declare -a MIRROR_DATA_DIRECTORY=(/data1/mirror
/data1/mirror /data1/mirror /data2/mirror /data2/mirror
/data2/mirror)
```

QD_PRIMARY_ARRAY, PRIMARY_ARRAY, MIRROR_ARRAY

These parameters can only be provided using an input configuration file, with the `gpinitsystem -I` *`input_configuration_file`* option. `QD_PRIMARY_ARRAY`, `PRIMARY_ARRAY`, and `MIRROR_ARRAY` define the Greenplum Database master host and the primary and mirror instances on the segment hosts, respectively, using the format:

```
host~port~data_directory/seg_prefix<segment_id>~dbid~content_id~replication_port
```

The Greenplum Database master always uses the value -1 for the segment ID and content ID. For example:

```
QD_PRIMARY_ARRAY=127.0.0.1~5432~/gpmaster/gpsne-1~1~-1~0
declare -a PRIMARY_ARRAY=(
127.0.0.1~40000~/gpdata1/gpsne0~2~0~6000
127.0.0.1~40001~/gpdata2/gpsne1~3~1~6001
)
```

```
declare -a MIRROR_ARRAY=(
127.0.0.1~50000~/gpmirror1/gpsne0~4~0~51000
127.0.0.1~50001~/gpmirror2/gpsne1~5~1~51001
)
```

You can use the `gpinitsystem -O` *`output_configuration_file`* to populate `QD_PRIMARY_ARRAY`, `PRIMARY_ARRAY`, `MIRROR_ARRAY` using the hosts, data directories, segment prefix, and base port values that you provide to the command. For recovery purposes, you can edit the segment and content IDs to match the values of an existing Greenplum Database backup.

Examples

Initialize a Greenplum Database array by supplying a cluster configuration file and a segment host address file, and set up a spread mirroring (`-S`) configuration:

```
$ gpinitsystem -c gpinitsystem_config -h
hostfile_gpinitsystem -S
```

Initialize a Greenplum Database array and set the superuser remote password:

```
$ gpinitsystem -c gpinitsystem_config -h
hostfile_gpinitsystem --su-password=mypassword
```

Initialize a Greenplum Database array with an optional standby master host:

```
$ gpinitsystem -c gpinitsystem_config -h
hostfile_gpinitsystem -s host09
```

Instead of initializing a Greenplum Database array, write the provided configuration to an output file. The output file uses the `QD_PRIMARY_ARRAY` and `PRIMARY_ARRAY` parameters to define master and segment hosts:

```
$ gpinitsystem -c gpinitsystem_config -h
hostfile_gpinitsystem -S -O cluster_init.config
```

Initialize a Greenplum Database using an input configuration file (a file that defines the Greenplum Database array using `QD_PRIMARY_ARRAY` and `PRIMARY_ARRAY` parameters:

```
$ gpinitsystem -I cluster_init.config
```

See Also

`gpssh-exkeys`, *`gpdeletesystem`*

gppkg

Installs Greenplum Database extensions such as pgcrypto, PL/R, PL/Java, PL/Perl, PostGIS, and MADlib, along with their dependencies, across an entire cluster.

Synopsis

```
gppkg [-i package | -u package | -r  name-version | -c]
        [-d master_data_directory] [-a] [-v]

gppkg --migrate GPHOME_old GPHOME_new [-a] [-v]

gppkg [-q | --query] query_option

gppkg -? | --help | -h

gppkg --version
```

Description

The Greenplum Package Manager (`gppkg`) utility installs Greenplum Database extensions, along with any dependencies, on all hosts across a cluster. It will also automatically install extensions on new hosts in the case of system expansion and segment recovery.

First, download one or more of the available packages from *Pivotal Network* then copy it to the master host. Use the Greenplum Package Manager to install each package using the options described below.

> **Note:** After a major upgrade to Greenplum Database, you must download and install all extensions again.

Examples of database extensions and packages software that are delivered using the Greenplum Package Manager are:

- PostGIS
- PL/Java
- PL/R
- PL/Perl
- MADlib
- Pgcrypto

Note that Greenplum Package Manager installation files for extension packages might release outside of standard product release cycles. Current packages are available from *Pivotal Network*. Information about package compatibility is the *Greenplum Database Release Notes*.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-c | --clean

Reconciles the package state of the cluster to match the state of the master host. Running this option after a failed or partial install/uninstall ensures that the package installation state is consistent across the cluster.

-d *master_data_directory*

The master data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

-i *package* | --install=*package*

Installs the given package. This includes any pre/post installation steps and installation of any dependencies.

--migrate *GPHOME_old GPHOME_new*

Migrates packages from a separate `$GPHOME`. Carries over packages from one version of Greenplum Database to another.

For example: `gppkg --migrate /usr/local/greenplum-db-4.3.14.0 /usr/local/greenplum-db-4.3.15.0`

When migrating packages, these requirements must be met.

- At least the master instance of the destination Greenplum Database must be started (the instance installed in *GPHOME_new*). Before running the `gppkg` command start the Greenplum Database master with the command `gpstart -m`.
- Run the `gppkg` utility from the *GPHOME_new* installation. The migration destination installation directory.

-q | --query *query_option*

Provides information specified by `query_option` about the installed packages. Only one `query_option` can be specified at a time. The following table lists the possible values for query_option. `<package_file>` is the name of a package.

Table 9: Query Options for gppkg

query_option	Returns
`<package_file>`	Whether the specified package is installed.
`--info <package_file>`	The name, version, and other information about the specified package.
`--list <package_file>`	The file contents of the specified package.
`--all`	List of all installed packages.

-r *name-version* | --remove=*name-version*

Removes the specified package.

-u *package* | --update=*package*

Updates the given package.

> **Warning:** The process of updating a package includes removing all previous versions of the system objects related to the package. For example, previous versions of shared libraries are removed. After the update process, a database function will fail when it is called if the function references a package file that has been removed.

--version (show utility version)

Displays the version of this utility.

-v | --verbose

Sets the logging level to verbose.

-? | -h | --help

Displays the online help.

gpscp

Copies files between multiple hosts at once.

Synopsis

```
gpscp { -f hostfile_gpssh | -h hostname [-h hostname ...] }
      [-J character] [-v] [[user@]hostname:]file_to_copy [...]
      [[user@]hostname:]copy_to_path

gpscp -?

gpscp --version
```

Description

The `gpscp` utility allows you to copy one or more files from the specified hosts to other specified hosts in one command using SCP (secure copy). For example, you can copy a file from the Greenplum Database master host to all of the segment hosts at the same time.

To specify the hosts involved in the SCP session, use the `-f` option to specify a file containing a list of host names, or use the `-h` option to name single host names on the command-line. At least one host name (`-h`) or a host file (`-f`) is required. The `-J` option allows you to specify a single character to substitute for the *hostname* in the `copy from` and `copy to` destination strings. If `-J` is not specified, the default substitution character is an equal sign (`=`). For example, the following command will copy `.bashrc` from the local host to `/home/gpadmin` on all hosts named in `hostfile_gpssh`:

```
gpscp -f hostfile_gpssh .bashrc =:/home/gpadmin
```

If a user name is not specified in the host list or with *user*`@` in the file path, `gpscp` will copy files as the currently logged in user. To determine the currently logged in user, do a `whoami` command. By default, `gpscp` goes to `$HOME` of the session user on the remote hosts after login. To ensure the file is copied to the correct location on the remote hosts, it is recommended that you use absolute paths.

Before using `gpscp`, you must have a trusted host setup between the hosts involved in the SCP session. You can use the utility `gpssh-exkeys` to update the known host files and exchange public keys between hosts if you have not done so already.

Options

-f *hostfile_gpssh*

Specifies the name of a file that contains a list of hosts that will participate in this SCP session. The syntax of the host file is one host per line as follows:

```
<hostname>
```

-h *hostname*

Specifies a single host name that will participate in this SCP session. You can use the `-h` option multiple times to specify multiple host names.

-J *character*

The `-J` option allows you to specify a single character to substitute for the *hostname* in the `copy from` and `copy to` destination strings. If `-J` is not specified, the default substitution character is an equal sign (`=`).

-v (verbose mode)

Optional. Reports additional messages in addition to the SCP command output.

file_to_copy

Required. The file name (or absolute path) of a file that you want to copy to other hosts (or file locations). This can be either a file on the local host or on another named host.

copy_to_path

Required. The path where you want the file(s) to be copied on the named hosts. If an absolute path is not used, the file will be copied relative to `$HOME` of the session user. You can also use the equal sign '`=`' (or another character that you specify with the `-J` option) in place of a *hostname*. This will then substitute in each host name as specified in the supplied host file (`-f`) or with the `-h` option.

-? (help)

Displays the online help.

--version

Displays the version of this utility.

Examples

Copy the file named `installer.tar` to `/` on all the hosts in the file `hostfile_gpssh`.

```
gpscp -f hostfile_gpssh installer.tar =:/
```

Copy the file named *myfuncs.so* to the specified location on the hosts named `sdw1` and `sdw2`:

```
gpscp -h sdw1 -h sdw2 myfuncs.so =:/usr/local/greenplum-db/lib
```

See Also

gpssh, *gpssh-exkeys*

gpseginstall

Installs Greenplum Database on segment hosts.

Synopsis

```
gpseginstall -f hostfile [-u gpdb_admin_user] [-p password]
             [-c u|p|c|s|E|e|l|v]

gpseginstall --help
```

Description

The `gpseginstall` utility provides a simple way to quickly install Greenplum Database on segment hosts that you specify in a host list file. The utility does not install or update Greenplum Database on the master host. You can run `gpseginstall` as `root` or as a non-root user. `gpseginstall` does not perform database initialization. See `gpinitsystem` for more information about initializing Greenplum Database.

When run as root, `gpseginstall` default actions are to add a system user (default is `gpadmin`), create a password (default is `changeme`), and deploy and install Greenplum Database on segment hosts. To do this, `gpseginstall` locates the current Greenplum Database binaries on the master from the installation path in the current user's environment variables (`$GPHOME`). It compresses Greenplum Database software into a `tar.gz` file and performs an MD5 checksum to verify file integrity.

Then, it copies Greenplum Database to the segment hosts, installs (decompresses) Greenplum Database, and changes the ownership of the Greenplum Database installation to the system user you specify with the `-u` option. Lastly, it exchanges keys between all Greenplum Database hosts as both root and as the system user you specify with the -u option. gpseginstall also perform a user limit check and verifies the version number of Greenplum Database on all the segments.

If you run gpseginstall as a non-root user, gpseginstall only compresses, copies, and installs Greenplum Database on segment hosts. It can also exchanges keys between Greenplum Database hosts for the current system user, and verifies the version number of Greenplum Database on all the segments.

Options

-c | --commands *option_list*

Optional. This allows you to customize `gpseginstall` actions. Note that these command options are executed by default if you do not specify the `-c` option in the `gpseginstall` syntax.

- `u`: Adds a system user. (`root` only)
- `p`: Changes the password for a system user. (`root` only)
- `s`: Compresses, copies, decompresses (installs) Greenplum Database on all segments.
- `c`: Changes the ownership of the Greenplum Database installation directory on the segment hosts. (`root` only)
- `E`: Exchange keys between Greenplum Database master and segment hosts for the root user. (`root` only)
- `e`: Exchange keys between Greenplum Database master and segment hosts for the non-root system user.
- `l`: (Linux only) Checks and modifies the user limits configuration file (`/etc/security/limits.conf` file) when adding a new user to segment hosts. (`root` only)
- `v`: Verifies the version of Greenplum Database running on all segments. `gpseginstall` checks the version number of the Greenplum Database installation referenced by the `$GPHOME` environment variable and symbolic link to the installation directory. An error

occurs if there is a version number mismatch or the Greenplum Database installation directory cannot be found.

-f | --file *hostfile*

Required. This specifies the file that lists the segment hosts onto which you want to install Greenplum Database.

The host list file must have one host name per line and includes a host name for each segment host in your Greenplum system. Make sure there are no blank lines or extra spaces. If a host has multiple configured host names, use only one host name per host. For example:

```
sdw1-1
sdw2-1
sdw3-1
sdw4-1
```

If available, you can use the same `gpssh-exkeys` host list file you used to exchange keys between Greenplum Database hosts.

-p | --password *password*

Optional. Sets the password for the user you specify with the `-u` option. The default password is `changeme`. This option is only available when you run `gpsetinstall` as root.

Recommended security best practices:

- Always use passwords.
- Do not use default passwords.
- Change default passwords immediately after installation.

-u | --user *user*

Optional. This specifies the system user. This user is also the Greenplum Database administrative user. This user owns Greenplum Database installation and administers the database. This is also the user under which Greenplum Database is started/initialized. This option is only available when you run `gpseginstall` as `root`. The default is `gpadmin`.

--help (help)

Displays the online help.

Examples

As `root`, install a Greenplum Database on all segments, leave the system user as the default (`gpadmin`) and set the `gpadmin` password to `secret123`:

```
# gpseginstall -f my_host_list_file -p secret123
```

As a non-root user, compress and copy Greenplum Database binaries to all segments (as `gpadmin`):

```
$ gpseginstall -f host_file
```

As root, add a user (`gpadmin2`), set the password for the user (`secret1234`), exchange keys between hosts as the new user, check user limits, and verify version numbers, but do not change ownership of Greenplum binaries, compress/copy/ install Greenplum Database on segments, or exchange keys as `root`.

```
$ gpseginstall -f host_file -u gpadmin2 -p secret1234 -c upelv
```

See Also

gpinitsystem, *gpssh-exkeys*

gpssh

Provides SSH access to multiple hosts at once.

Synopsis

```
gpssh { -f hostfile_gpssh | - h hostname [-h hostname ...] } [-s] [-e]
      [-d seconds] [-t multiplier] [-v]
      [bash_command]

gpssh -?

gpssh --version
```

Description

The `gpssh` utility allows you to run bash shell commands on multiple hosts at once using SSH (secure shell). You can execute a single command by specifying it on the command-line, or omit the command to enter into an interactive command-line session.

To specify the hosts involved in the SSH session, use the `-f` option to specify a file containing a list of host names, or use the `-h` option to name single host names on the command-line. At least one host name (`-h`) or a host file (`-f`) is required. Note that the current host is ***not*** included in the session by default — to include the local host, you must explicitly declare it in the list of hosts involved in the session.

Before using `gpssh`, you must have a trusted host setup between the hosts involved in the SSH session. You can use the utility `gpssh-exkeys` to update the known host files and exchange public keys between hosts if you have not done so already.

If you do not specify a command on the command-line, `gpssh` will go into interactive mode. At the `gpssh` command prompt (`=>`), you can enter a command as you would in a regular bash terminal command-line, and the command will be executed on all hosts involved in the session. To end an interactive session, press `CTRL+D` on the keyboard or type `exit` or `quit`.

If a user name is not specified in the host file, `gpssh` will execute commands as the currently logged in user. To determine the currently logged in user, do a `whoami` command. By default, `gpssh` goes to `$HOME` of the session user on the remote hosts after login. To ensure commands are executed correctly on all remote hosts, you should always enter absolute paths.

If you encounter network timeout problems when using `gpssh`, you can use `-d` and `-t` options or set parameters in the `gpssh.conf` file to control the timing that `gpssh` uses when validating the initial `ssh` connection. For information about the configuration file, see *gpssh Configuration File*.

Options

bash_command

A bash shell command to execute on all hosts involved in this session (optionally enclosed in quotes). If not specified, `gpssh` starts an interactive session.

-d (delay) *seconds*

Optional. Specifies the time, in seconds, to wait at the start of a `gpssh` interaction with `ssh`. Default is `0.05`. This option overrides the `delaybeforesend` value that is specified in the `gpssh.conf` configuration file.

Increasing this value can cause a long wait time during `gpssh` startup.

-e (echo)

Optional. Echoes the commands passed to each host and their resulting output while running in non-interactive mode.

-f *hostfile_gpssh*

Specifies the name of a file that contains a list of hosts that will participate in this SSH session. The syntax of the host file is one host per line.

-h *hostname*

Specifies a single host name that will participate in this SSH session. You can use the `-h` option multiple times to specify multiple host names.

-s

Optional. If specified, before executing any commands on the target host, `gpssh` sources the file `greenplum_path.sh` in the directory specified by the `$GPHOME` environment variable.

This option is valid for both interactive mode and single command mode.

-t *multiplier*

Optional. A decimal number greater than 0 (zero) that is the multiplier for the timeout that `gpssh` uses when validating the `ssh` prompt. Default is `1`. This option overrides the `prompt_validation_timeout` value that is specified in the `gpssh.conf` configuration file.

Increasing this value has a small impact during `gpssh` startup.

-v (verbose mode)

Optional. Reports additional messages in addition to the command output when running in non-interactive mode.

--version

Displays the version of this utility.

-? (help)

Displays the online help.

gpssh Configuration File

The `gpssh.conf` file contains parameters that let you adjust the timing that `gpssh` uses when validating the initial `ssh` connection. These parameters affect the network connection before the `gpssh` session executes commands with `ssh`. The location of the file is specified by the environment variable `MASTER_DATA_DIRECTORY`. If the environment variable is not defined or the `gpssh.conf` file does not exist, `gpssh` uses the default values or the values set with the `-d` and `-t` options. For information about the environment variable, see the *Greenplum Database Reference Guide*.

The `gpssh.conf` file is a text file that consists of a `[gpssh]` section and parameters. On a line, the `#` (pound sign) indicates the start of a comment. This is an example `gpssh.conf` file.

```
[gpssh]
delaybeforesend = 0.05
prompt_validation_timeout = 1.0
sync_retries = 5
```

These are the `gpssh.conf` parameters.

delaybeforesend = *seconds*

Specifies the time, in seconds, to wait at the start of a `gpssh` interaction with `ssh`. Default is 0.05. Increasing this value can cause a long wait time during `gpssh` startup. The `-d` option overrides this parameter.

prompt_validation_timeout = *multiplier*

A decimal number greater than 0 (zero) that is the multiplier for the timeout that `gpssh` uses when validating the `ssh` prompt. Increasing this value has a small impact during `gpssh` startup. Default is `1`. The `-t` option overrides this parameter.

sync_retries = *attempts*

A non-negative integer that specifies the maximum number of times that `gpssh` attempts to connect to a remote Greenplum Database host. The default is 3. If the value is 0, `gpssh` returns an error if the initial connection attempt fails. Increasing the number of attempts also increases the time between retry attempts. This parameter cannot be configured with a command-line option.

The `-t` option also affects the time between retry attempts.

Increasing this value can compensate for slow network performance or segment host performance issues such as heavy CPU or I/O load. However, when a connection cannot be established, an increased value also increases the delay when an error is retuned.

Examples

Start an interactive group SSH session with all hosts listed in the file `hostfile_gpssh`:

```
$ gpssh -f hostfile_gpssh
```

At the `gpssh` interactive command prompt, run a shell command on all the hosts involved in this session.

```
=> ls -a /data/primary/*
```

Exit an interactive session:

```
=> exit
=> quit
```

Start a non-interactive group SSH session with the hosts named `sdw1` and `sdw2` and pass a file containing several commands named `command_file` to `gpssh`:

```
$ gpssh -h sdw1 -h sdw2 -v -e < command_file
```

Execute single commands in non-interactive mode on hosts `sdw2` and `localhost`:

```
$ gpssh -h sdw2 -h localhost -v -e 'ls -a /data/primary/*'
$ gpssh -h sdw2 -h localhost -v -e 'echo $GPHOME'
$ gpssh -h sdw2 -h localhost -v -e 'ls -1 | wc -l'
```

See Also

gpssh-exkeys, *gpscp*

gpssh-exkeys

Exchanges SSH public keys between hosts.

Synopsis

```
gpssh-exkeys -f hostfile_exkeys | - h hostname [-h hostname ...]

gpssh-exkeys -e hostfile_exkeys -x hostfile_gpexpand

gpssh-exkeys -?

gpssh-exkeys --version
```

Description

The `gpssh-exkeys` utility exchanges SSH keys between the specified host names (or host addresses). This allows SSH connections between Greenplum hosts and network interfaces without a password prompt. The utility is used to initially prepare a Greenplum Database system for password-free SSH access, and also to add additional ssh keys when expanding a Greenplum Database system.

To specify the hosts involved in an initial SSH key exchange, use the `-f` option to specify a file containing a list of host names (recommended), or use the `-h` option to name single host names on the command-line. At least one host name (`-h`) or a host file is required. Note that the local host is included in the key exchange by default.

To specify new expansion hosts to be added to an existing Greenplum Database system, use the `-e` and `-x` options. The `-e` option specifies a file containing a list of existing hosts in the system that already have SSH keys. The `-x` option specifies a file containing a list of new hosts that need to participate in the SSH key exchange.

Keys are exchanged as the currently logged in user. You should perform the key exchange process twice: once as `root` and once as the `gpadmin` user (the user designated to own your Greenplum Database installation). The Greenplum Database management utilities require that the same non-root user be created on all hosts in the Greenplum Database system, and the utilities must be able to connect as that user to all hosts without a password prompt.

The `gpssh-exkeys` utility performs key exchange using the following steps:

- Creates an RSA identification key pair for the current user if one does not already exist. The public key of this pair is added to the `authorized_keys` file of the current user.
- Updates the `known_hosts` file of the current user with the host key of each host specified using the `-h`, `-f`, `-e`, and `-x` options.
- Connects to each host using `ssh` and obtains the `authorized_keys`, `known_hosts`, and `id_rsa.pub` files to set up password-free access.
- Adds keys from the `id_rsa.pub` files obtained from each host to the `authorized_keys` file of the current user.
- Updates the `authorized_keys`, `known_hosts`, and `id_rsa.pub` files on all hosts with new host information (if any).

Options

-e *hostfile_exkeys*

When doing a system expansion, this is the name and location of a file containing all configured host names and host addresses (interface names) for each host in your *current* Greenplum system (master, standby master and segments), one name per line without blank

lines or extra spaces. Hosts specified in this file cannot be specified in the host file used with `-x`.

-f *hostfile_exkeys*

Specifies the name and location of a file containing all configured host names and host addresses (interface names) for each host in your Greenplum system (master, standby master and segments), one name per line without blank lines or extra spaces.

-h *hostname*

Specifies a single host name (or host address) that will participate in the SSH key exchange. You can use the `-h` option multiple times to specify multiple host names and host addresses.

--version

Displays the version of this utility.

-x *hostfile_gpexpand*

When doing a system expansion, this is the name and location of a file containing all configured host names and host addresses (interface names) for each *new segment host* you are adding to your Greenplum system, one name per line without blank lines or extra spaces. Hosts specified in this file cannot be specified in the host file used with `-e`.

-? (help)

Displays the online help.

Examples

Exchange SSH keys between all host names and addresses listed in the file `hostfile_exkeys`:

```
$ gpssh-exkeys -f hostfile_exkeys
```

Exchange SSH keys between the hosts `sdw1`, `sdw2`, and `sdw3`:

```
$ gpssh-exkeys -h sdw1 -h sdw2 -h sdw3
```

Exchange SSH keys between existing hosts `sdw1`, `sdw2`, and `sdw3`, and new hosts `sdw4` and `sdw5` as part of a system expansion operation:

```
$ cat hostfile_exkeys
mdw
mdw-1
mdw-2
smdw
smdw-1
smdw-2
sdw1
sdw1-1
sdw1-2
sdw2
sdw2-1
sdw2-2
sdw3
sdw3-1
sdw3-2
$ cat hostfile_gpexpand
sdw4
sdw4-1
sdw4-2
sdw5
sdw5-1
sdw5-2
$ gpssh-exkeys -e hostfile_exkeys -x hostfile_gpexpand
```

See Also

gpssh, *gpscp*

gpstart

Starts a Greenplum Database system.

Synopsis

```
gpstart [-d master_data_directory] [-B parallel_processes] [-R]
        [-m] [-y] [-a] [-t timeout_seconds] [-l logfile_directory]
        [-v | -q]

gpstart -? | -h | --help

gpstart --version
```

Description

The `gpstart` utility is used to start the Greenplum Database server processes. When you start a Greenplum Database system, you are actually starting several `postgres` database server listener processes at once (the master and all of the segment instances). The `gpstart` utility handles the startup of the individual instances. Each instance is started in parallel.

The first time an administrator runs `gpstart`, the utility creates a hosts cache file named `.gphostcache` in the user's home directory. Subsequently, the utility uses this list of hosts to start the system more efficiently. If new hosts are added to the system, you must manually remove this file from the `gpadmin` user's home directory. The utility will create a new hosts cache file at the next startup.

Before you can start a Greenplum Database system, you must have initialized the system using `gpinitsystem` first.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-B *parallel_processes*

The number of segments to start in parallel. If not specified, the utility will start up to 64 parallel processes depending on how many segment instances it needs to start.

-d *master_data_directory*

Optional. The master host data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-m (master only)

Optional. Starts the master instance only, which may be useful for maintenance tasks. This mode only allows connections to the master in utility mode. For example:

```
PGOPTIONS='-c gp_session_role=utility' psql
```

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-R (restricted mode)

Starts Greenplum Database in restricted mode (only database superusers are allowed to connect).

-t *timeout_seconds*

Specifies a timeout in seconds to wait for a segment instance to start up. If a segment instance was shutdown abnormally (due to power failure or killing its `postgres` database listener process, for example), it may take longer to start up due to the database recovery and validation process. If not specified, the default timeout is 60 seconds.

-v (verbose output)

Displays detailed status, progress and error messages output by the utility.

-y (do not start standby master)

Optional. Do not start the standby master host. The default is to start the standby master host and synchronization process.

-? | -h | --help (help)

Displays the online help.

--version (show utility version)

Displays the version of this utility.

Examples

Start a Greenplum Database system:

```
gpstart
```

Start a Greenplum Database system in restricted mode (only allow superuser connections):

```
gpstart -R
```

Start the Greenplum master instance only and connect in utility mode:

```
gpstart -m PGOPTIONS='-c gp_session_role=utility' psql
```

See Also

gpstop, *gpinitsystem*

gpstop

Stops or restarts a Greenplum Database system.

Synopsis

```
gpstop [-d master_data_directory] [-B parallel_processes]
       [-M smart | fast | immediate] [-t timeout_seconds] [-r] [-y] [-a]
       [-l logfile_directory] [-v | -q]

gpstop -m [-d master_data_directory] [-y] [-l logfile_directory] [-v | -q]

gpstop -u [-d master_data_directory] [-l logfile_directory] [-v | -q]

gpstop --host host_name [-d master_data_directory] [-l logfile_directory]
       [-t timeout_seconds] [-a] [-v | -q]

gpstop --version

gpstop -? | -h | --help
```

Description

The `gpstop` utility is used to stop the database servers that comprise a Greenplum Database system. When you stop a Greenplum Database system, you are actually stopping several `postgres` database server processes at once (the master and all of the segment instances). The `gpstop` utility handles the shutdown of the individual instances. Each instance is shutdown in parallel.

By default, you are not allowed to shut down Greenplum Database if there are any client connections to the database. Use the `-M fast` option to roll back all in progress transactions and terminate any connections before shutting down. If there are any transactions in progress, the default behavior is to wait for them to commit before shutting down.

With the `-u` option, the utility uploads changes made to the master `pg_hba.conf` file or to *runtime* configuration parameters in the master `postgresql.conf` file without interruption of service. Note that any active sessions will not pickup the changes until they reconnect to the database.

Options

-a (do not prompt)

Do not prompt the user for confirmation.

-B *parallel_processes*

The number of segments to stop in parallel. If not specified, the utility will start up to 64 parallel processes depending on how many segment instances it needs to stop.

-d *master_data_directory*

Optional. The master host data directory. If not specified, the value set for `$MASTER_DATA_DIRECTORY` will be used.

--host *host_name*

The utility shuts down the Greenplum Database segment instances on the specified host to allow maintenance on the host. Each primary segment instance on the host is shut down and the associated mirror segment instance is promoted to a primary segment if the mirror segment is on another host. Mirror segment instances on the host are shut down.

The segment instances are not shut down and the utility returns an error in these cases:

- Segment mirroring is not enabled for the system.

- The master or standby master is on the host.
- Both a primary segment instance and its mirror are on the host.

This option cannot be specified with the `-m`, `-r`, `-u`, or `-y` options.

> **Note:** The `gprecoverseg` utility restores segment instances. Run `gprecoverseg` commands to start the segments as mirrors and then to return the segments to their preferred role (primary segments).

-l *logfile_directory*

The directory to write the log file. Defaults to `~/gpAdminLogs`.

-m (master only)

Optional. Shuts down a Greenplum master instance that was started in maintenance mode.

-M fast (fast shutdown - rollback)

Fast shut down. Any transactions in progress are interrupted and rolled back.

-M immediate (immediate shutdown - abort)

Immediate shut down. Any transactions in progress are aborted.

This mode kills all `postgres` processes without allowing the database server to complete transaction processing or clean up any temporary or in-process work files.

-M smart (smart shutdown - warn)

Smart shut down. If there are active connections, this command fails with a warning. This is the default shutdown mode.

-q (no screen output)

Run in quiet mode. Command output is not displayed on the screen, but is still written to the log file.

-r (restart)

Restart after shutdown is complete.

-t *timeout_seconds*

Specifies a timeout threshold (in seconds) to wait for a segment instance to shutdown. If a segment instance does not shutdown in the specified number of seconds, `gpstop` displays a message indicating that one or more segments are still in the process of shutting down and that you cannot restart Greenplum Database until the segment instance(s) are stopped. This option is useful in situations where `gpstop` is executed and there are very large transactions that need to rollback. These large transactions can take over a minute to rollback and surpass the default timeout period of 600 seconds.

-u (reload `pg_hba.conf` and `postgresql.conf` files only)

This option reloads the `pg_hba.conf` files of the master and segments and the runtime parameters of the `postgresql.conf` files but does not shutdown the Greenplum Database array. Use this option to make new configuration settings active after editing `postgresql.conf` or `pg_hba.conf`. Note that this only applies to configuration parameters that are designated as *runtime* parameters.

-v (verbose output)

Displays detailed status, progress and error messages output by the utility.

-y (do not stop standby master)

Do not stop the standby master process. The default is to stop the standby master.

-? | -h | --help (help)

Displays the online help.

--version (show utility version)

Displays the version of this utility.

Examples

Stop a Greenplum Database system in smart mode:

```
gpstop
```

Stop a Greenplum Database system in fast mode:

```
gpstop -M fast
```

Stop all segment instances and then restart the system:

```
gpstop -r
```

Stop a master instance that was started in maintenance mode:

```
gpstop -m
```

Reload the `postgresql.conf` and `pg_hba.conf` files after making configuration changes but do not shutdown the Greenplum Database array:

```
gpstop -u
```

See Also

gpstart

gptransfer

The `gptransfer` utility copies objects from databases in a source Greenplum Database system to databases in a destination Greenplum Database system.

Synopsis

```
gptransfer
   { --full |
   { [-d database1 [ -d database2 ... ]] |
   [-t db.schema.table [ -t db.schema1.table1 ... ]] |
   [-f table-file [--partition-transfer
     | --partition-transfer-non-partition-target ]]
   [-T db.schema.table [ -T db.schema1.table1 ... ]]
   [-F table-file] } }
   [--skip-existing | --truncate | --drop]
   [--analyze] [--validate=type] [-x] [--dry-run]
   [--schema-only ]
   [--source-host=source_host [--source-port=source_port]
   [--source-user=source_user]]
   [--base-port=base_gpfdist_port]
   [--dest-host=dest_host --source-map-file=host_map_file
   [--dest-port=port] [--dest-user=dest_user] ]
   [--dest-database=dest_database_name]
   [--batch-size=batch_size] [--sub-batch-size=sub_batch_size]
   [--timeout=seconds]
   [--max-line-length=length]
   [--work-base-dir=work_dir] [-l log_dir]
   [--format=[CSV|TEXT] ]
   [--quote=character ]
   [--no-final-count ]

   [-v | --verbose]
   [-q | --quiet]
   [--gpfdist-verbose]
   [--gpfdist-very-verbose]
   [-a]

gptransfer --version

gptransfer -h | -? | --help
```

Description

The `gptransfer` utility copies database objects from a source Greenplum Database system to a destination system. You can perform one of the following types of operations:

- Copy a Greenplum Database system with the `--full` option.

  This option copies all user created databases in a source system to a different destination system. If you specify the `--full` option, you must specify both a source and destination system. The destination system cannot contain any user-defined databases, only the default databases `postgres`, `template0`, and `template1`.
- Copy a set of user defined database tables to a destination system. The `-f`, and `-t` options copy a specified set of user defined tables, table data, and re-creates the table indexes. The `-d` option copies all user defined tables, table data, and re-creates the table indexes from a specified database.

  If the destination system is the same as the source system, you must also specify a destination database with the `--dest-database` option. When you specify a destination database, the source database tables are copied into the specified destination database.

For partitioned tables, you can specify the `--partition-transfer` or the `--partition-transfer-non-partition-target` option with `-f` option to copy specific leaf child partitions of partitioned tables from a source database. The leaf child partitions are the lowest level partitions of a partitioned database. For the `--partition-transfer` option, the destination tables are leaf child partitions. For the `--partition-transfer-non-partition-target` option, the destination tables are non-partitioned tables.

If an invalid set of `gptransfer` options are specified, or if a specified source table or database does not exist, `gptransfer` returns an error and quits. No data is copied.

To copy database objects between Greenplum Database systems `gptransfer` utility uses:

- The Greenplum Database utility `gpfdist` on the source database system. The `gpfdists` protocol is not supported.
- Writable external tables on the source database system and readable external tables on the destination database system. Note that `gptransfer` creates external tables with the `execute` protocol, so the `--source_user` account must have superuser privileges (the default is `gpadmin`).
- Named pipes that transfer the data between a writable external table and a readable external table.

When copying data into the destination system, it is redistributed on the Greenplum Database segments of the destination system. This is the flow of data when `gptransfer` copies database data:

writable external table > gpfdist > named pipe > gpfdist > readable external table

For information about transferring data with `gptransfer`, see "Migrating Data with Gptransfer" in the *Greenplum Database Administrator Guide*.

About Database, Schema, and Table Names

When you transfer an entire database, schema and table names in the database can contain only alphanumeric characters and the underscore character (_). Also, Unicode characters are not supported. The same naming restrictions apply when you specify a database, schema, or table as an option or in a file.

Notes

The `gptransfer` utility efficiently transfers tables with large amounts of data. Because of the overhead required to set up parallel transfers, the utility is not recommended for transferring tables with small amounts of data. It might be more efficient to copy the schema and smaller tables to the destination database using other methods, such as the SQL `COPY` command, and then use `gptransfer` to transfer large tables in batches.

The `gptransfer` utility can copy a data row with a maximum length of 256 MB.

When transferring data between databases, you can run only one instance of `gptransfer` at a time. Running multiple, concurrent instances of `gptransfer` is not supported.

When copying database data between different Greenplum Database systems, `gptransfer` requires a text file that lists all the source segment host names and IP addresses. Specify the name and location of the file with the `--source-map-file` option. If the file is missing or not all segment hosts are listed, `gptransfer` returns an error and quits. See the description of the option for file format information.

The source and destination Greenplum Database segment hosts need to be able to communicate with each other. To ensure that the segment hosts can communicate, you can use a tool such as the Linux `netperf` utility.

If a filespace has been created for a source Greenplum Database system, a corresponding filespace must exist on the target system.

SSH keys must be exchanged between the two systems before using `gptransfer`. The `gptransfer` utility connects to the source system with SSH to create the named pipes and start the `gpfdist` instances. You

can use the Greenplum Database `gpssh-exkeys` utility with a list of all the source and destination primary hosts to exchange keys between Greenplum Database hosts.

Source and destination systems must be able to access the `gptransfer` work directory. The default directory is the user's home directory. You can specify a different directory with the `--work-base-dir` option.

The `gptransfer` utility logs messages in the `~/gpAdminLogs` directory on the master host. `gptransfer` creates a log file with the name `gptransfer_`*`date`*`.log` and appends messages to it each time it runs on that day. You can specify a different directory for the log file using the `-l` *`log_directory`* option.

Work directories named `~/gptransfer_`*`process_id`* are created on segment hosts in the source cluster. Log files for the `gpfdist` instances that `gptransfer` creates are in these directories. Adding the `--gpfdist-verbose` or `--gpfdist-very-verbose` options to the `gptransfer` command line increases the `gpfdist` logging level.

The `gptransfer` utility does not move configuration files such as `postgres.conf` and `pg_hba.conf`. You must set up the destination system configuration separately.

The `gptransfer` utility does not move external objects such as Greenplum Database extensions, third party jar files, and shared object files. You must install the external objects separately.

The `gptransfer` utility does not move dependent database objects unless you specify the `--full` option. For example, if a table has a default value on a column that is a user-defined function, that function must exist in the destination system database when using the `-t`, `-d`, or `-f` options.

If you move a set of database tables with the `-d`, `-t`, or `-f` option, and the destination table or database does not exist, `gptransfer` creates it. The utility re-creates any indexes on tables before copying data.

If a table exists on the destination system and one of the options `--skip-existing`, `--truncate`, or `--drop` is not specified, `gptransfer` returns an error and quits.

If an error occurs when during the process of copying a table, or table validation fails, `gptransfer` continues copying the other specified tables. After `gptransfer` finishes, it displays a list of tables where an error occurred, writes the names of tables that failed into a text file, and then prints the name of the file. You can use this file with the `gptransfer` -f option to retry copying tables.

The name of the file that contains the list of tables where errors occurred is `failed_migrated_tables_`*`yyyymmdd_hhmmss`*`.txt`. The *yyyymmdd_hhmmss* is a time stamp when the `gptransfer` process was started. The file is created in the directory were `gptransfer` is executed.

After `gptransfer` completes copying database objects, the utility compares the row count of each table copied to the destination databases with the table in the source database. The utility returns the validation results for each table. You can disable the table row count validation by specifying the `--no-final-count` option.

> **Note:** If the number of rows do not match, the table is not added to the file that lists the tables where transfer errors occurred.

The `gp_external_max_segs` server configuration parameter controls the number of segment instances that can access a single `gpfdist` instance simultaneously. Setting a low value might affect `gptransfer` performance. For information about the parameter, see the *Greenplum Database Reference Guide*.

Limitation for the Source and Destination Systems

If you are copying data from a system with a larger number of segments to a system with a fewer number of segment hosts, then the total number of primary segments on the destination system must be greater than or equal to the total number of segment hosts on the source system.

For example, a quarter rack V1 DCA has a total of 24 primary segments. The means the source side cannot have more than 24 segment hosts (one and one-half racks).

When you copy data from a source Greenplum Database system with a larger number of primary segment instances than on the destination system, the data transfer might be slower when compared to a transfer

where the source system has fewer segment instances than the destination system. The `gptransfer` utility uses a different configuration of named pipes and `gpfdist` instances in the two situations.

Options

-a

Quiet mode, do not prompt the user for confirmation.

--analyze

Run the `ANALYZE` command on non-system tables. The default is to not run the `ANALYZE` command.

--base-port=*base_gpfdist_port*

Base port for `gpfdist` on source segment systems. If not specified, the default is 8000.

--batch-size=*batch_size*

Sets the maximum number of tables that `gptransfer` concurrently copies to the destination database. If not specified, the default is 2. The maximum is 10.

> **Note:** If the order of the transfer is important, specify a value of 1. The tables are transferred sequentially based on the order specified in the `-t` and `-f` options.

-d *database*

A source database to copy. This option can be specified multiple times to copy multiple databases to the destination system. All the user defined tables and table data are copied to the destination system.

A set of databases can be specified using the Python regular expression syntax. The regular expression pattern must be enclosed in slashes (`/RE_pattern/`). If you use a regular expression, the name must be enclosed in double quotes ("). This example `-d "demo/.*/"` specifies all databases in the Greenplum Database installation that begin with `demo`.

> **Note:** Note the following two examples for the `-d` option are equivalent. They both specify a set of databases that begins with `demo` and ends with zero or more digits.
>
> ```
> -d "demo/[0-9]*/"
> -d "/demo[0-9]*/"
> ```

If the source database does not exist, `gptransfer` returns an error and quits. If a destination database does not exist a database is created.

Not valid with the `--full`, `-f`, `-t`, `--partition-transfer`, or `--partition-transfer-non-partition-target` options.

Alternatively, specify the `-t` or `-f` option to copy a specified set of tables.

--delimiter=*delim*

Delimiter to use for writable external tables created by `gptransfer`. Specify a single ASCII character that separates columns within each row of data. The default value is a comma (`,`). If *delim* is a comma (`,`) or if this option is not specified, `gptransfer` uses the `CSV` format for writable external tables. Otherwise, `gptransfer` uses the `TEXT` format.

If `--delimiter`, `--format`, and `--quote` options are not specified, these are settings for writable external tables:

```
FORMAT 'CSV' ( DELIMITER ',' QUOTE E'\001' )
```

You can specify a delimiter character such as a non-printing character with the format "`\digits`" (octal). A backslash followed by the octal value for the character. The octal format

must be enclosed in double quotes. This example specifies the octal character `\001`, the `SOH` character:

```
--delimiter="\001"
```

--dest-database=*dest_database_name*

The database in the destination Greenplum Database system. If not specified, the source tables are copied into a destination system database with the same name as the source system database.

This option is required if the source and destination Greenplum Database systems are the same.

If destination database does not exist, it is created.

Not valid with the `--full`, `--partition-transfer`, or `--partition-transfer-non-partition-target` options.

--dest-host=*dest_host*

Destination Greenplum Database hostname or IP address. If not specified, the default is the host the system running `gptransfer` (127.0.0.1)

--dest-port=*dest_port*

Destination Greenplum Database port number, If not specified, the default is 5432.

--dest-user=*dest_user*

User ID that is used to connect to the destination Greenplum Database system. If not specified, the default is the user gpadmin.

--drop

Specify this option to drop the table that is in the destination database if it already exists. Before copying table data, `gptransfer` drops the table and creates it again.

At most, only one of the options can be specified `--skip-existing`, `--truncate`, or `--drop`. If one of them is not specified and the table exists in the destination system, `gptransfer` returns an error and quits.

Not valid with the `--full`, `--partition-transfer`, or `--partition-transfer-non-partition-target` options.

--dry-run

When you specify this option, `gptransfer` generates a list of the migration operations that would have been performed with the specified options. The data is not migrated.

The information is displayed at the command line and written to the log file.

-f *table-file*

The location and name of file containing list of fully qualified table names to copy from the Greenplum Database source system. In the text file, you specify a single fully qualified table per line (*database*.*schema*.*table*).

A set of tables can be specified using the Python regular expression syntax. See the `-d` option for information about using regular expressions.

If the source table does not exist, `gptransfer` returns an error and quits. If the destination database or table does not exist, it is created.

Only the table and table data are copied and indexes are re-created. Dependent objects are not copied.

You cannot specify views, or system catalog tables. The `--full` option copies user defined views.

If you specify the `-d` option to copy all the tables from a database, you cannot specify individual tables from the database.

Not valid with the `--full`, `-d`, or `-t` options.

--partition-transfer (partitioned destination table)

Specify this option with the `-f` option to copy data from leaf child partition tables of partitioned tables from a source database to the leaf child partition tables in a destination database. The text file specified by the `-f` option contains a list of fully qualified leaf child partition table names with this syntax.

```
src_db.src_schema.src_prt_tbl[, dst_db.dst_schema.dst_prt_tbl]
```

Wildcard characters are not supported in the fully qualified table names. The destination partitioned table must exist. If the destination leaf child partition table is not specified in the file, `gptransfer` copies the data to the same fully qualified table name (*db_name*.*schema*.*table*) in the destination Greenplum Database system. If the source and destination Greenplum Database systems are the same, you must specify a destination table where at least one of the following must be different between the source and destination table: *db_name*, *schema*, or *table*.

If either the source or destination table is not a leaf child partition, the utility returns an error and no data are transferred.

These characteristics must be the same for the partitioned table in the source and destination database.

- Number of table columns and the order of the column data types (the source and destination table names and table column names can be different)
- Partition level of the specified source and destination tables
- Partitioning criteria of the specified source and destination leaf child partitions and child partitions above them in the hierarchy (partition type and partition column)

This option is not valid with these options: `-d`, `--dest-database`, `--drop`, `-F`, `--full`, `--schema-only`, `-T`, `-t`.

> **Note:** If a destination table is not empty or the data in the source or destination table changes during a transfer operation (rows are inserted or deleted), the table row count validation fails due to row count mismatch.
>
> If the destination table is not empty, you can specify the `-truncate` option to truncate the table before the transfer operation.
>
> You can specify the `-x` option to acquire exclusive locks on the tables during a transfer operation.

--partition-transfer-non-partition-target (non-partitioned destination table)

Specify this option with the `-f` option to copy data from leaf child partition tables of partitioned tables in a source database to non-partitioned tables in a destination database. The text file specified by the `-f` option contains a list of fully qualified leaf child partition table names in the source database and non-partitioned tables names in the destination database with this syntax.

```
src_db.src_schema.src_part_tbl, dest_db.dest_schema.dest_tbl
```

Wildcard characters are not supported in the fully qualified table names. The destination tables must exist, and both source and destination table names are required in the file.

If a source table is not a leaf child partition table or a destination table is not a normal (non-partitioned) table, the utility returns an error and no data are transferred.

If the source and destination Greenplum Database systems are the same, you must specify a destination table where at least one of the following must be different between the source and destination table: *db_name*, *schema*, or *table*.

For the partitioned table in the source database and the table in the destination database, the number of table columns and the order of the column data types must be the same (the source and destination table column names can be different).

The same destination table can be specified in the file for multiple source leaf child partition tables that belong to a single partitioned table. Transferring data from source leaf child partition tables that belong to different partitioned tables to a single non-partitioned table is not supported.

This option is not valid with these options: `-d`, `--dest-database`, `--drop`, `-F`, `--full`, `--schema-only`, `-T`, `-t`, `--truncate`, `--validate`.

Note: If the data in the source or destination table changes during a transfer operation (rows are inserted or deleted), the table row count validation fails due to row count mismatch.

You can specify the `-x` option to acquire exclusive locks on the tables during a transfer operation.

-F *table-file*

The location and name of file containing list of fully qualified table names to exclude from transferring to the destination system. In the text file, you specify a single fully qualified table per line.

A set of tables can be specified using the Python regular expression syntax. See the `-d` option for information about using regular expressions.

The utility removes the excluded tables from the list of tables that are being transferred to the destination database before starting the transfer. If excluding tables results in no tables being transferred, the database or schema is not created in the destination system.

If a source table does not exist, `gptransfer` displays a warning.

Only the specified tables are excluded. To exclude dependent objects, you must explicitly specify them.

You cannot specify views, or system catalog tables.

Not valid with the `--full`, `--partition-transfer`, or `--partition-transfer-non-partition-target` options.

You can specify the `--dry-run` option to test the command. The `-v` option, displays and logs the excluded tables.

--format=[CSV | TEXT]

Specify the format of the writable external tables that are created by `gptransfer` to transfer data. Values are `CSV` for comma separated values, or `TEXT` for plain text. The default value is `CSV`.

If the options `--delimiter`, `--format`, and `--quote` are not specified, these are default settings for writable external tables:

```
FORMAT 'CSV' ( DELIMITER ',' QUOTE E'\001' )
```

If you specify `TEXT`, you must also specify a non-comma delimiter with the `--delimiter=`*`delim`* option. These are settings for writable external tables:

```
FORMAT 'TEXT' ( DELIMITER delim ESCAPE 'off' )
```

--full

Full migration of a Greenplum Database source system to a destination system. You must specify the options for the destination system, the `--source-map-file` option, the `--dest-host` option, and if necessary, the other destination system options.

The `--full` option cannot be specified with the `-t`, `-d`, `-f`, `--partition-transfer`, or `--partition-transfer-non-partition-target` options.

A full migration copies all database objects including, tables, indexes, views, users, roles, functions, and resource queues for all user defined databases. The default databases, postgres, template0 and template1 are not moved.

If a database exists in the destination system, besides the default postgres, template0 and template1 databases, `gptransfer` returns an error and quits.

> **Note:** The `--full` option is recommended only when the databases contain a large number of tables with large amounts of data. Because of the overhead required to set up parallel transfers, the utility is not recommended when the databases contain tables with small amounts of data. For more information, see *Notes*.

--gpfdist-verbose

Set the logging level for `gpfdist` processes to verbose (`-v`). Cannot be specified with `--gpfdist-very-verbose`.

The output is recorded in `gpfdist` log files in the `~/gptransfer_`*`process_id`* directory on segment hosts in the source Greenplum Database cluster.

--gpfdist-very-verbose

Set the logging level for `gpfdist` processes to very verbose (`-V`). Cannot be specified with `--gpfdist-verbose`.

The output is recorded in `gpfdist` log files in the `~/gptransfer_`*`process_id`* directory on segment hosts in the source Greenplum Database cluster.

-l *log_dir*

Specify the `gptransfer` log file directory. If not specified, the default is `~/gpAdminLogs`. This directory is created on the master host in the source Greenplum cluster.

--max-line-length=*length*

Sets the maximum allowed data row length in bytes for the `gpfdist` utility. If not specified, the default is 10485760. Valid range is 32768 (32K) to 268435456 (256MB).

Should be used when user data includes very wide rows (or when `line too long` error message occurs). Should not be used otherwise as it increases resource allocation.

--no-final-count

Disable table row count validation that is performed after `gptransfer` completes copying database objects to the target database. The default is to compare the row count of tables copied to the destination databases with the tables in the source database.

-q | --quiet

If specified, suppress status messages. Messages are only sent to the log file.

--quote=*character*

The quotation character when `gptransfer` creates writable external tables with the `CSV` format. Specify a single ASCII character that is used to enclose column data. The default value is the octal character `\001`, the `SOH` character.

You can specify a delimiter character such as a non-printing character with the format "`\`*`digits`*" (octal). A backslash followed by the octal value for the character. The octal value must be enclosed in double quotes.

--schema-only

Create only the schemas specified by the command. Data is not transferred.

If specified with the `--full` option, `gptransfer` replicates the complete database schema, including all tables, indexes, views, user defined types (UDT), and user defined functions (UDF) for the source databases. No data is transferred.

If you specify databases with the `-d` option or tables with the `-t` or `-f` options, `gptransfer` creates only the tables and indexes. No data is transferred.

Not valid with the `--partition-transfer`, `--partition-transfer-non-partition-target`, or `--truncate` options.

> **Note:** Because of the overhead required to set up parallel transfers, the `--schema-only` option is not recommended when transferring information for a large number of tables. For more information, see *Notes*.

--skip-existing

Specify this option to skip copying a table from the source database if the table already exists in the destination database.

At most, only one of the options can be specified `--skip-existing`, `--truncate`, or `--drop`. If one of them is not specified and the table exists in the destination system, `gptransfer` returns an error and quits.

Not valid with the `--full` option.

--source-host=*source_host*

Source Greenplum Database host name or IP address. If not specified, the default host is the system running `gptransfer` (127.0.0.1).

--source-map-file=*host_map_file*

File that lists source segment host name and IP addresses. If the file is missing or not all segment hosts are listed, `gptransfer` returns an error and quits.

Each line of the file contains a source host name and the host IP address separated by a comma: `hostname,IPaddress`. This example lists four Greenplum Database hosts and their IP addresses.

```
sdw1,192.0.2.1
sdw2,192.0.2.2
sdw3,192.0.2.3
sdw4,192.0.2.4
```

This option is required if the `--full` option is specified or if the source Greenplum Database system is different than the destination system. This option is not required if source and destination systems are the same.

--source-port=*source_port*

Source Greenplum Database port number. If not specified, the default is 5432.

--source-user=*source_user*

User ID that is used to connect to the source Greenplum Database system. If not specified, the default is the user gpadmin.

> **Note:** `gptransfer` creates external tables with the `execute` protocol, so the `--source_user` that you specify must have superuser privileges.

--sub-batch-size=*sub_batch_size*

Specifies the maximum degree of parallelism of the operations performed when migrating a table such as starting gpfdist instances, creating named pipes for the move operations. If not specified, the default is 25. The maximum is 50.

Specify the `--batch-size` option to control the maximum number of tables that `gptransfer` concurrently processes.

-t *db.schema.table*

A table from the source database system to copy. The fully qualified table name must be specified.

A set of tables can be specified using the Python regular expression syntax. See the `-d` option for information about using regular expressions.

If the destination table or database does not exist, it is created. This option can be specified multiple times to include multiple tables. Only the table and table data are copied and indexes are re-created. Dependent objects are not copied.

If the source table does not exist, `gptransfer` returns an error and quits.

If you specify the `-d` option to copy all the tables from a database, you do not need to specify individual tables from the database.

Not valid with the `--full`, `-d`, `-f`, `--partition-transfer`, or `--partition-transfer-non-partition-target` options.

-T *db.schema.table*

A table from the source database system to exclude from transfer. The fully qualified table name must be specified.

A set of tables can be specified using the Python regular expression syntax. See the `-d` option for information about using regular expressions.

This option can be specified multiple times to include multiple tables. Only the specified tables are excluded. To exclude dependent objects, you must explicitly specify them.

The utility removes the excluded tables from the list of tables that are being transferred to the destination database before starting the transfer. If excluding tables results in no tables being transferred, the database or schema is not created in the destination system.

If a source table does not exist, `gptransfer` displays a warning.

Not valid with the `--full`, `--partition-transfer`, or `--partition-transfer-non-partition-target`options.

You can specify the `--dry-run` option to test the command. The `-v` option displays and logs the excluded tables.

--timeout *seconds*

Specify the time out value in seconds that `gptransfer` passes the `gpfdist` processes that `gptransfer` uses. The value is the time allowed for Greenplum Database to establish a connection to a `gpfdist` process. You might need to increase this value when operating on high-traffic networks.

The default value is 300 seconds (5 minutes). The minimum value is 2 seconds, the maximum value is 600 seconds.

--truncate

Specify this option to truncate the table that is in the destination database if it already exists.

At most, only one of the options can be specified `--skip-existing`, `--truncate`, or `--drop`. If one of them is not specified and the table exists in the destination system, `gptransfer` returns an error and quits.

Not valid with the `--full` option.

--validate=*type*

Perform data validation on table data. These are the supported types of validation.

`count` - Specify this value to compare row counts between source and destination table data.

`MD5` - Specify this value to compare MD5 values between source and destination table data.

Note: On Greenplum Database systems with FIPS enabled, the option `md5` is not supported. Use the `count` option to validate table data.

If validation for a table fails, `gptransfer` displays the name of the table and writes the file name to the text file `failed_migrated_tables_`*yyyymmdd_hhmmss*`.txt`. The *yyyymmdd_hhmmss* is a time stamp when the `gptransfer` process was started. The file is created in the directory where `gptransfer` is executed.

Note: The file contains the table names where validation failed or other errors occurred during table migration.

-v | --verbose

If specified, sets the logging level to verbose. Additional log information is written to the log file and the command line during command execution.

--work-base-dir=*work_dir*

Specify the directory that `gptransfer` uses to store temporary working files such as PID files and named pipes. The default directory is the user's home directory.

Source and destination systems must be able to access the `gptransfer` work directory.

-x

Acquire an exclusive lock on tables during the migration to prevent insert or updates.

On the source database, an exclusive lock is acquired when `gptransfer` inserts into the external table and is released after validation.

On the destination database, an exclusive lock is acquired when `gptransfer` selects from external table and released after validation.

If `-x` option is not specified and `--validate` is specified, validation failures occur if data is inserted into either the source or destination table during the migration process. The `gptransfer` utility displays messages if validation errors occur.

-h | -? | --help

Displays the online help.

--version

Displays the version of this utility.

Examples

This command copies the table `public.t1` from the database `db1` and all tables in the database `db2` to the system `mytest2`.

```
gptransfer -t db1.public.t1 -d db2 --dest-host=mytest2 \
 --source-map-file=gp-source-hosts --truncate
```

If the databases `db1` and `db2` do not exist on the system `mytest2`, they are created. If any of the source tables exist on the destination system, `gptransfer` truncates the table and copies the data from the source to the destination table.

This command copies leaf child partition tables from a source system to a destination system.

```
gptransfer -f input_file --partition-transfer --source-host=source_host \
 --source-user=source_user --source-port=source_port --dest-host=dest_host \
 --dest-user=dest_user --dest-port=dest_port --source-map-file=host_map_file
```

This line in `input_file` copies a leaf child partition from the source system to the destination system.

```
 srcdb.people.person_1_prt_experienced, destdb.public.employee_1_prt_seniors
```

The line assumes partitioned tables in the source and destination systems similar to the following tables.

- In the *people* schema of the *srcdb* database of the source system, a partitioned table with a leaf child partition table `person_1_prt_experienced`. This `CREATE TABLE` command creates a partitioned table with the leaf child partition table.

```
CREATE TABLE person(id int, title char(1))
  DISTRIBUTED BY (id)
  PARTITION BY list (title)
  (PARTITION experienced VALUES ('S'),
    PARTITION entry_level VALUES ('J'),
    DEFAULT PARTITION other );
```

- In the *public* schema of the *destdb* database of the source system, a partitioned table with a leaf child partition table `public.employee_1_prt_seniors`. This `CREATE TABLE` command creates a partitioned table with the leaf child partition table.

```
CREATE TABLE employee(id int, level char(1))
  DISTRIBUTED BY (id)
  PARTITION BY list (level)
  (PARTITION seniors VALUES ('S'),
    PARTITION juniors VALUES ('J'),
    DEFAULT PARTITION other );
```

This example uses Python regular expressions in a filter file to specify the set of tables to transfer. This command specifies the `-f` option with the filter file `/tmp/filter_file` to limit the tables that are transferred.

```
gptransfer -f /tmp/filter_file --source-port 5432 --source-host test4 \
 --source-user gpadmin --dest-user gpadmin --dest-port 5432 --dest-host test1 \
 --source-map-file /home/gpadmin/source_map_file
```

This is the contents of `/tmp/filter_file`.

```
"test1.arc/.*/./.*/"
"test1.c/(..)/y./.*/"
```

In the first line, the regular expressions for the schemas, `arc/.*/`, and for the tables, `/.*/`, limit the transfer to all tables with the schema names that start with `arc`.

In the second line, the regular expressions for the schemas, `c/(..)/y`, and for the tables, `/.*/`, limit the transfer to all tables with the schema names that are four characters long and that start with `c` and end with `y`, for example, `crty`.

When the command is run, tables in the database `test1` that satisfy either condition are transferred to the destination database.

See Also

gpfdist

For information about loading and unloading data, see the *Greenplum Database Administrator Guide*.

Chapter 10

Greenplum Environment Variables

This reference lists and describes the environment variables to set for Greenplum Database. Set these in your user's startup shell profile (such as `~/.bashrc` or `~/.bash_profile`), or in `/etc/profile` if you want to set them for all users.

Required Environment Variables

Note: `GPHOME`, `PATH` and `LD_LIBRARY_PATH` can be set by sourcing the `greenplum_path.sh` file from your Greenplum Database installation directory

GPHOME

This is the installed location of your Greenplum Database software. For example:

```
GPHOME=/usr/local/greenplum-db-4.3.x.x
export GPHOME
```

PATH

Your `PATH` environment variable should point to the location of the Greenplum Database `bin` directory. For example:

```
PATH=$GPHOME/bin:$PATH
export PATH
```

LD_LIBRARY_PATH

The `LD_LIBRARY_PATH` environment variable should point to the location of the Greenplum Database/ PostgreSQL library files. For example:

```
LD_LIBRARY_PATH=$GPHOME/lib
export LD_LIBRARY_PATH
```

MASTER_DATA_DIRECTORY

This should point to the directory created by the gpinitsystem utility in the master data directory location. For example:

```
MASTER_DATA_DIRECTORY=/data/master/gpseg-1
export MASTER_DATA_DIRECTORY
```

Optional Environment Variables

The following are standard PostgreSQL environment variables, which are also recognized in Greenplum Database. You may want to add the connection-related environment variables to your profile for convenience, so you do not have to type so many options on the command line for client connections. Note that these environment variables should be set on the Greenplum Database master host only.

PGAPPNAME

The name of the application that is usually set by an application when it connects to the server. This name is displayed in the activity view and in log entries. The `PGAPPNAME` environmental variable behaves the same as the `application_name` connection parameter. The default value for `application_name` is `psql`. The name cannot be longer than 63 characters.

PGDATABASE

The name of the default database to use when connecting.

PGHOST

The Greenplum Database master host name.

PGHOSTADDR

The numeric IP address of the master host. This can be set instead of or in addition to `PGHOST` to avoid DNS lookup overhead.

PGPASSWORD

The password used if the server demands password authentication. Use of this environment variable is not recommended for security reasons (some operating systems allow non-root users to see process environment variables via `ps`). Instead consider using the `~/.pgpass` file.

PGPASSFILE

The name of the password file to use for lookups. If not set, it defaults to `~/.pgpass`. See the topic about *The Password File* in the PostgreSQL documentation for more information.

PGOPTIONS

Sets additional configuration parameters for the Greenplum Database master server.

PGPORT

The port number of the Greenplum Database server on the master host. The default port is 5432.

PGUSER

The Greenplum Database user name used to connect.

PGDATESTYLE

Sets the default style of date/time representation for a session. (Equivalent to `SET datestyle TO...`)

PGTZ

Sets the default time zone for a session. (Equivalent to `SET timezone TO...`)

PGCLIENTENCODING

Sets the default client character set encoding for a session. (Equivalent to `SET client_encoding TO...`)