

VMware Tanzu Greenplum Command Center v6.0 Documentation

VMware Tanzu Greenplum Command Center 6.0

You can find the most up-to-date technical documentation on the VMware website at:
<https://docs.vmware.com/>

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Copyright © 2022 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

Contents

Pivotal Greenplum Command Center 6.0.0 Documentation	11
Pivotal Greenplum® Command Center 6.0.0 Release Notes	13
About This Release	13
Supported Platforms	13
Enhancements and Changes in Greenplum Command Center 6.0	13
Command Center Installation Changes	13
Transitioning to the gpmetrics History Schema	14
Other Improvements	14
Fixed Issues	15
Known Issues	15
Command Center Installer Truncates pg_hba.conf LDAP Entries	15
Failure to Auto-Create Monthly Partitions in gpmetrics Schema	15
Workaround	16
Cyrillic Font Support in the Pivotal UI	17
Unable to View Real Time Queries After Upgrading From a Previous Command Center Release	17
Resource Queue Name Incorrect for Some Queries	17
Cannot Set SSH Path When Upgrading Command Center	17
About Pivotal Greenplum Command Center	18
Greenplum Command Center Features	18
Greenplum Command Center Architecture	18
Greenplum Command Center Web Server and Web Application	19
Greenplum Database Performance Monitoring Database	20
Real-Time Query Metrics Collection	20
Command Center Workload Management	21
Installing Greenplum Command Center	22
Installing Pivotal Greenplum Command Center	22
Prerequisites	22
Selecting and Preparing an Installation Directory for Command Center	23
Install the Greenplum Command Center Software	23
Interactive Installation	24
Install With a Configuration File	26

Non-Interactive Installation with Defaults	27
Upgrade	27
Prepare the Standby Master Host	28
Next Steps	28
Upgrading Greenplum Command Center	29
Uninstalling Greenplum Command Center	30
Setting the Greenplum Command Center Environment	31
Connecting to the Greenplum Command Center Console	32
Administering the Command Center Web Server	34
Starting and Stopping the Web Server	34
Configuring the Command Center Web Server	35
Viewing and Maintaining Web Server Log Files	35
Administering Command Center Agents	37
About the gpperfmon Data Collection Agents	37
Configuring gpmon Role Logging Options	37
Adding and Removing Hosts	38
Viewing and Maintaining Agent Log Files	38
Administering the gpperfmon Database	39
Connecting to the gpperfmon Database	39
Backing Up and Restoring the gpperfmon Database	39
Maintaining the Historical Data Tables	39
Settings	40
History settings	40
gpcc	42
Actions	42
Options	43
Description	43
Examples	43
gpmetrics Schema Reference	44
Alert Tables	44
gpcc_alert_rule Table	44
gpcc_alert_log Table	45

Example Query	45
Greenplum Database Metrics History Tables	46
gpcc_database_history	46
gpcc_disk_history	46
gpcc_plannode_history	47
gpcc_queries_history	48
gpcc_system_history	50
Configuration Files Reference	52
Command Center Console Parameters	52
gpmetrics Configuration File Reference	53
Setup Configuration File	54
Parameters	54
Examples	55
Securing Greenplum Command Center	57
SSL/TLS Encryption	57
Authentication Options	57
Authorization	58
Managing Greenplum Command Center Authentication	59
Viewing the Host-Based Authentication File	60
Editing the Host-Based Authentication File	60
Loading a Previous Version of the Host-Based Authentication File	61
Managing Greenplum Command Center Permissions	61
Viewing User Permissions	62
Changing User Permission Levels	62
Enabling or Disabling Guest Access to Query Monitor	63
Securing the gpmon Database User	63
Changing the gpmon Password	63
Authenticating gpmon with Kerberos	64
Enabling Authentication with Kerberos	65
Before You Begin	65
Add Command Center Principals to the KDC Database	66
Adding Kerberos Principals	67

Set Up Keytab Files	67
Command Center Running on the Greenplum Master Host	67
Command Center Running on the Standby Master	68
Update the Greenplum Database pg_hba.conf File	69
Enable Kerberos for Command Center	70
Authenticating With Kerberos on the Client Workstation	70
Monitoring the Greenplum Database System	71
Dashboard	71
System Information	71
Segment Health	72
Database State	72
Segment Status	72
Disk Usage Summary	73
Queries	73
CPU	73
Memory	73
Alerts	73
Greenplum Database Cluster State	73
System Information	74
Segment Health	74
Database State	74
Segment Status	75
Disk Usage Summary	75
Queries	75
CPU	75
Memory	75
Alerts	75
Segment Status	76
Segment Summary	76
Segment Health	77
Preferred Roles	77
Segment Table	78
Cluster Metrics	78
Host Metrics	80
About Skew Calculations	81

Storage Status	81
Disk Usage Summary	81
GP Segments Usage History	82
GP Masters Usage History	82
Monitoring and Managing Greenplum Database Queries	83
Query Monitor	83
Query Metrics	84
Using the Query Monitor Controls	85
Query Details	85
Query Execution Metrics	86
Query Text and Execution Plan	88
Query Text	88
Query Plan and Progress	88
Textual Plan	90
History	91
Query Metrics	92
Query History Details	93
Query History Metrics	94
Query Text and Execution Plan	95
Query Text	95
Query Plan and Progress	95
Textual Plan	96
Alerts	98
Configuring Alert Rules	98
Configuring Alert Email	100
Configuring email With Command Center	100
Configuring email for Greenplum Database	101
Creating a Send Alert Script	102
Managing Greenplum Database Workloads	104
About Greenplum Command Center Workload Management	104
About Query Assignment Rules	105
Workload Management	105

Defining Resource Groups and Resource Attributes	105
Assigning Roles to Resource Groups	107
Defining Workload Management Rules	107
Defining and Setting Query Tags	108
Importing Resource Queues to Resource Groups	109
Step One: Enable Resource Groups in Greenplum Database	109
Step Two: Preview and Configure Resource Group Imports	109
Step Three: Enable Command Center Workload Management	111
Accessing the Workload Configuration Programmatically	111
Workload Management Rules JSON Format	112
version pair	112
assignmentRules array	112
gpmetrics.get_workload_config()	113
Example	113
gpmetrics.set_workload_config()	114
Example	114
Troubleshooting Enabling Resource Groups	114
Query Monitor Help Topics	118
CPU	118
CPU Skew	118
What is CPU Skew?	118
Data Skew	119
Computational Skew	119
Locks	119
Memory	121
Resource Group Memory Configuration Parameters	121
Greenplum Database Memory Configuration Parameters	122
Query Plan Execution	122
Slices and Gangs	123
Operators	123
Scan Operators	123
Join Operators	125
Sort and Aggregate Operators	125

Motion Operators	125
DML Operators	125
Spill Files	125
Investigating Spill File Usage	126
Eliminating or Reducing Spill Files	126
Limiting Spill Files with Server Configuration Parameters	126
Alert Help Topics	128
Database connectivity failure	128
What is this alert?	128
What to do?	128
Segment failure	129
What is this alert?	129
What to do	129
Average memory (segment hosts) exceeds [%] for [min]	129
What is this alert?	129
What to do?	129
Memory (master) exceeds [%] for [min]	130
What is this alert?	130
What to do	130
Total disk space exceeds [%]	130
What is this alert?	130
What to do	130
Query is blocked for [min]	131
What is this alert?	131
Alert details	131
What to do	131
Number of connections exceeds [n]	131
What is this alert?	131
What to do	131
Look for idle or stale connections and terminate them	131
CPU (master) exceeds [%] for [min]	132
What is this alert?	132

What to do	132
See also	132
Out of memory error	133
What is this alert?	133
What to do	133
Query runtime exceeds [min]	133
What is this alert?	133
Alert details	133
What to do	133
Average CPU (segment hosts) exceeds [%] for [min]	134
What is this alert?	134
What to do	134
Spill files for a query exceeds [GB]	134
What is this alert?	134
What to do	134
For more information	134

Pivotal Greenplum Command Center 6.0.0 Documentation

PDF

[Greenplum Command Center 6.0.0 Release Notes](#)

[About Greenplum Command Center](#)

Installing Greenplum Command Center

- [Installing Greenplum Command Center](#)
- [Setting the Greenplum Command Center Environment](#)

Administration

- [Administering the Command Center Web Server](#)
- [Administering Command Center Agents](#)
- [Administering the gpperfmon Database](#)
- [Administering the gpmetrics Query History](#)
- [gpcc Command Reference](#)
- [Configuration File Reference](#)
- [Securing Greenplum Command Center](#)

Using Greenplum Command Center

- [Monitoring the Greenplum Database System](#)
 - ◊ [Overall Cluster State](#)
 - ◊ [Segment Status](#)
 - ◊ [Cluster Metrics](#)
 - ◊ [Host Metrics](#)
 - ◊ [Storage Status](#)
- [Monitoring and Managing Greenplum Database Queries](#)
 - ◊ [Query Monitor](#)
 - ◊ [Query Details](#)
 - ◊ [Query History](#)
- [Managing Alerts](#)
- [Managing Workloads](#)

- ◊ [About Workloads](#)
- ◊ [Managing Workloads with Command Center](#)
- ◊ [Troubleshooting Command Center Workload Management](#)

Command Center Help Topics

- [CPU](#)
- [CPU Skew](#)
- [Locks](#)
- [Memory](#)
- [Query Optimization](#)
- [Spill Files](#)

Pivotal Greenplum® Command Center 6.0.0 Release Notes

Updated 9/30/2019

About This Release

This document contains release information about Pivotal Greenplum Command Center 6.0. Greenplum Command Center 6.0 provides management and monitoring functionality for Pivotal Greenplum Database 6.

See [Enhancements and Changes in Greenplum Command Center 6.0.0](#) for information about new features and changes in this Command Center release.

Supported Platforms

Greenplum Command Center 6.0.0 is compatible with the following platforms.

- Pivotal Greenplum Database 6.0
- Red Hat Enterprise Linux 6.x¹ and 7.x
- CentOS 6.x¹ and 7.x
- Ubuntu 18.04

Enhancements and Changes in Greenplum Command Center 6.0

- The **Enable GPCC history data collection** option on the **Admin> Settings** page is now on by default. If you turned off this option in your current Command Center installation and you upgrade to Command Center 6.0.0, the option will remain off. Pivotal recommends that you turn it on.
- Command Center can collect all metrics without the Greenplum Database gpperfmon service. For lower overhead, stop the gpperfmon service if it is running. Set the `gp_enable_gpperfmon` server configuration parameter to off and restart Greenplum Database.

Command Center Installation Changes

- Installing Pivotal Greenplum Database 6 includes the Command Center metrics collector extension and workload manager extension. The `gp_enable_query_metrics` server configuration parameter is set to `on` and the metrics collector shared library is added to the `shared_preload_libraries` configuration parameter.

- It is no longer required to run `gpperfmon_install` to create the gpperfmon database before installing Command Center. The Command Center installer creates the gpperfmon database, the gpmon role, and the gpmetrics schema, if they do not already exist, and adds entries for the gpmon role to the `pg_hba.conf` file.
Note that the Command Center installer *does not* create the old gpperfmon tables or set the `gp_enable_gpperfmon` configuration parameter to `on`, so the `gpmmmon` and `gpsmon` data collection agents will not be started when you start Greenplum Database. If you want to use the old gpperfmon tables and agents, use the `gpperfmon_install` utility to create the gpperfmon database before you install Command Center. See [Transitioning to the gpmetrics History Schema](#) for more information about transitioning to the new gpmetrics history data.
- If you include the `-w` option with the `gpccinstall` command, the installer prompts you to enter the password for the gpmon role. If the gpmon role does not already exist, the installer creates it with the password you specify. Without the `-w` option, the installer creates the gpmon role with the default password “changeme”.
- The installer has a new `-u` (upgrade) option that installs Command Center using the configuration parameters from the current Command Center installation. You can use this option to reinstall the same version of Command Center or to upgrade to a newer version.
- Japanese and Russian language options have been added to the Command Center installer.
- The Command Center gpmon superuser creates external web tables, and this requires the Greenplum Database `gp_external_enable_exec` configuration parameter to be set to `on`. The parameter is on by default, but if it has been set to `off`, the Command Center installer will print a message and quit.

Transitioning to the gpmetrics History Schema

There can be two sets of history tables, both maintained in the gpperfmon database:

- tables created by the Greenplum Database gpperfmon installation utility belonging to the public schema
- tables created by Command Center belonging to the gpmetrics schema

Command Center 6 only requires the gpmetrics schema and it displays query and system metrics history from the gpmetrics schema by default. Before Command Center 6, both sets of history tables were required and Command Center displayed history from the tables in the public schema by default.

You can maintain both sets of tables if you choose, but for best performance, use only the gpmetrics schema.

To use both sets of tables, you must run the Greenplum Database `gpperfmon_install` utility to create the gpperfmon database before you install Command Center. To use only the gpmetrics history, let the Command Center installer create the gpperfmon database. You can disable the gpperfmon agents by setting the `gp_enable_gpperfmon` parameter to `off` and restarting Greenplum Database.

Other Improvements

- New filter options are added to the **History** view to select query history by:

- ◊ completion status
 - ◊ resource queue/priority or resource group
 - ◊ CPU skew greater than 0
 - ◊ Spill files greater than 0
- Historical query search performance has been greatly improved.
 - Partitions for history tables in the `gpmetrics` schema are now created with RLE compression and the `COMPRESSLEVEL=2` option, for improved compression/performance.
 - The Command Center web server now supports only the TLSv1.2 protocol and above for secured connections.
 - When canceling one or more queries in the Command Center UI, a pop-up box prompts you to enter a message of up to 128 characters to add to the error message that is displayed to the user.
 - On the **System> Storage Status** view, when you expand a **Hostname**, the **Data Directory** column now lists the mount points of partitions on the segment host file system that contain Greenplum segment data directories. When you move your cursor over a mount point, a tooltip lists the data directories the partition contains.
 - Logging improvements. A new `log_level` parameter in the `$GPCC_HOME/conf/app.conf` configuration file determines the level of messages that will be logged. The parameter specifies which messages are added to these log files: `gpccinstall.log`, `webserver.log`, `agent.log`, and `cli.log`.

The parameter can be set to `Debug`, `Info`, or `Error`, where `Debug` is the most verbose and `Error` is the least verbose. The parameter values are not case-sensitive. The default is `Info`.

Fixed Issues

The following issues are fixed in this release.

- [#166346339] If CPUSET is enabled for a resource group, the CPU% value is changed to `-1`. This value prevented editing resource groups in Command Center. Command Center now allows a `-1` value in the CPU% field.
- [#167156751] Query text files in the `$MASTER_DATA_DIRECTORY/gpmetrics/query_text` directory were not cleaned up automatically. This issue is fixed.

Known Issues

The following are known issues in the current Greenplum Command Center release.

Command Center Installer Truncates `pg_hba.conf` LDAP Entries

When the Command Center `gpccinstall` installer updates the `pg_hba.conf` file it truncates LDAP information on entries that use the `ldap` authentication method. The truncated information must be reinserted manually after the Command Center installation is complete. This issue exists in Command Center versions 4.7 through 4.9, 6.0, and 6.1.

Failure to Auto-Create Monthly Partitions in gpmetrics Schema

In time zones with daylight savings time (DST) ending in the month of October, Greenplum Command Center fails to create new October partitions for tables in the gpmetrics schema because it specifies the same start and end date for the partitions. The error does not occur in time zones that do not have DST, or in time zones with DST ending in November.

Workaround

Manually create partitions for October 2019.

Log in to the Greenplum master server as the gpadmin user and connect to the gpperfmon database as the gpmon user:

```
psql -U gpmon gpperfmon
```

Enter the following commands to create the October 2019 partitions:

```
ALTER TABLE gpmetrics.gpcc_disk_history
  ADD PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
  WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=RLE_TYPE, COMPRESSLEVEL=2);

ALTER TABLE gpmetrics.gpcc_system_history
  ADD PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
  WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=RLE_TYPE, COMPRESSLEVEL=2);

ALTER TABLE gpmetrics.gpcc_queries_history
  ADD PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
  WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=RLE_TYPE, COMPRESSLEVEL=2);

ALTER TABLE gpmetrics.gpcc_plannode_history
  ADD PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
  WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=RLE_TYPE, COMPRESSLEVEL=2);

ALTER TABLE gpmetrics.gpcc_database_history
  ADD PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
  WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=RLE_TYPE, COMPRESSLEVEL=2);

ALTER TABLE gpmetrics.gpcc_alert_log
  ADD PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
  WITH (APPENDONLY=true, ORIENTATION=column, COMPRESSTYPE=RLE_TYPE, COMPRESSLEVEL=2);

ALTER TABLE gpmetrics.gpcc_pg_log_history
  SPLIT DEFAULT PARTITION START (date '2019-10-01') INCLUSIVE END (date '2019-11-01') EXCLUSIVE
```



```
INTO (PARTITION t2019Oct, DEFAULT PARTITION);
```

Cyrillic Font Support in the Pivotal UI

The Source Sans Pro font in the Pivotal UI toolkit does not contain Cyrillic symbols, so the Command Center UI falls back to the default sans-serif font. This may cause some cosmetic rendering problems for Russian users.

Unable to View Real Time Queries After Upgrading From a Previous Command Center Release

If you install a new version of Greenplum Command Center using the same port number as the previous version, and you use the Chrome web browser, you may be unable to view real-time queries until after you clear the browser's cache. See the note in [Connecting to the Command Center Console](#) for steps to clear the browser cache.

Resource Queue Name Incorrect for Some Queries

To display the resource queue name for queries in the Query Monitor, Command Center caches user IDs with resource queue names. The cache is updated when the Command Center backend receives a planned query from the metrics collector. Some utility commands are not planned, such as `COPY` commands that have no `SELECT` clause. These commands use a resource queue slot, but are not passed through an optimizer. If a user is assigned a different resource queue and then executes an unplanned command, the Command Center Query Monitor will display the resource queue name that was previously cached.

Cannot Set SSH Path When Upgrading Command Center

The Command Center installer `-ssh_path <path>` command-line option specifies the path to a custom SSH command and also saves the path to the `ssh_path` parameter of the `app.conf` configuration file. If both the `-u` (upgrade) and `-ssh_path` options are specified on the `gpccinstall` command line, however, the `-ssh_path` option is ignored. The installer does not use the custom SSH command and the path is not saved in the `app.conf` file.

About Pivotal Greenplum Command Center

Pivotal Greenplum Command Center is a management tool for the Pivotal Greenplum Database Big Data Platform. This topic introduces key concepts about Greenplum Command Center and its components.

Greenplum Command Center Features

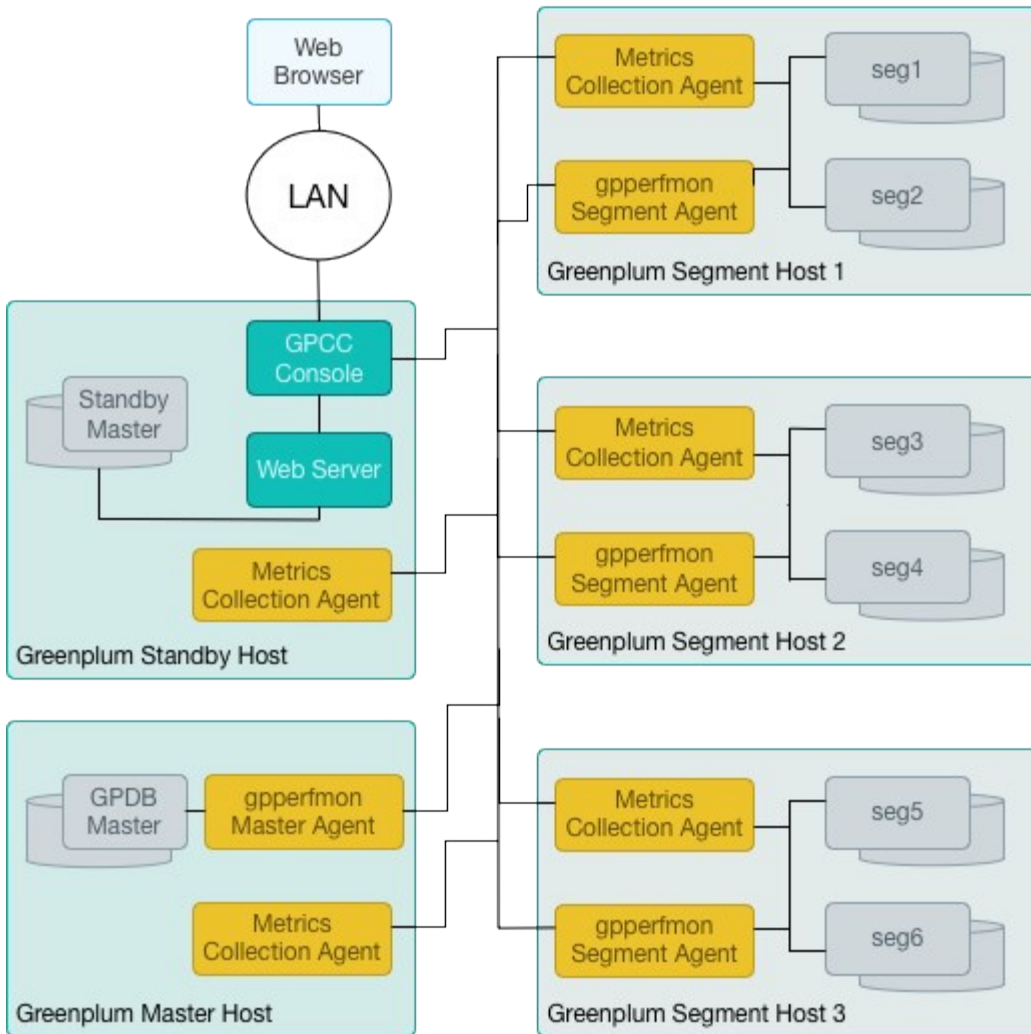
Greenplum Command Center monitors system performance metrics, analyzes cluster health, and enables database administrators to perform management tasks in a Greenplum Database environment.

Greenplum Command Center provides a browser-native HTML5 graphical console for viewing Greenplum Database system metrics and performing certain database administrative tasks. The Command Center application provides the following functionality:

- Interactive overview of realtime system metrics. Drill down to see details for individual cluster hosts and segments.
- Detailed realtime statistics for the cluster and by server.
- Query Monitor view lists queries executing, waiting to execute, and blocked by locks held by other queries.
- Query Detail view shows query metrics, query text, and the execution plan for the query.
- Workload Management view allows administrators to:
 - ◊ Create and manage workloads to manage concurrency and allocate CPU and memory resources.
 - ◊ Change default resource groups for Greenplum Database roles.
 - ◊ Create assignment rules to assign transactions to resource groups.
- Four permission levels allow users to view or cancel their own or others' queries, and to view or manage administrative information.
- Cluster Metrics view shows synchronized charts of historical system metrics.
- History view lists completed queries and system metrics plotted over a selected time period.
- Permissions view to see or manage Command Center permission levels.
- Authentication view to see or edit the `pg_hba.conf` host-based authentication configuration file.
- Segment Status view with summaries and details by segment.
- Storage Status view with summaries and details by segment data directory.

Greenplum Command Center Architecture

The following figure illustrates the Greenplum Command Center architecture.



Greenplum Command Center Web Server and Web Application

The Greenplum Command Center web server and backend application can run on the master or standby master host—the master host is recommended. The web server, `gppccws`, is a custom HTTP server designed for Command Center. The web application is an HTML5 and Go language application.

The Command Center web server authenticates users with the Greenplum Database authentication system. Administrators can edit the Greenplum Database host-based authentication file, `pg_hba.conf`, in the Command Center Console. Command Center can also be configured to authenticate users in a Kerberos environment.

Command Center defines four user authorization levels to manage users' access to the Query Monitor, and to administrative information and operations. User authorization is managed in the Administrative area of the Command Center user interface.

Greenplum Command Center displays information derived from several sources:

- Greenplum Database performance monitoring database (`gpperfmon`)
- Operating system process accounting

- Greenplum Database system catalog tables
- Real-time query metrics collection extension
- Workload management extension

Greenplum Database is instrumented to enable capturing performance metrics and tracking query execution. The performance monitoring database and the query metrics collection extension deploy agents—processes running on each host to collect metrics. The `gpperfmon` agents forward collected data to an agent on the Greenplum Database master. The real-time query metrics agents submit collected data directly to the Command Center `rpc` port. The agents also collect data from the host operating system so that query performance can be correlated with CPU and memory utilization and disk space can be monitored in Command Center.

Greenplum Database Performance Monitoring Database

The `gpperfmon` performance monitoring database stores current and historical query status and system information collected from agents running on the master and segment hosts. Greenplum Command Center uses `gpperfmon` for historical data only; it uses the real-time query metrics to monitor active and queued queries. Greenplum Database sends UDP packets at various points during query execution. The `gpsmon` process on each segment host collects the data. Periodically, every 15 seconds by default, a `gpmmmon` agent on the master host signals the `gpsmon` process to forward the collected data. The agent on the master host receives the data and adds it to the `gpperfmon` database.

The Command Center database consists of three sets of tables:

- *now* tables store data on current system metrics such as active queries
- *history* tables store data on historical metrics
- *tail* tables are for data in transition. Tail tables are for internal use only and should not be queried by users.

The *now* and *tail* data are stored as text files on the master host file system, and the Command Center database accesses them via external tables. The *history* tables are regular database tables stored within the `gpperfmon` database.

You can run SQL queries on the data stored in the `gpperfmon` database. Greenplum Command Center runs queries on the database for information presented in the Command Center Console. The *Greenplum Database Reference Guide* contains references for the tables in the `gpperfmon` database.

Greenplum Database provides a management utility, `gpperfmon_install`, to create the `gpperfmon` database and enable the `gpperfmon` agents on the master and segment hosts. Creating the `gpperfmon` database is a prerequisite for installing Greenplum Command Center. See the *Greenplum Database Utility Guide* for details of running the `gpperfmon_install` management utility.

Real-Time Query Metrics Collection

The data collected by real-time query metrics collection is more detailed and more current than statistics recorded in the `gpperfmon` database. Command Center users can observe queries as they execute and, with sufficient permissions, cancel problem queries to allow other queries to complete.

The Greenplum Database query metrics extension and the metrics collection agent work together to collect real-time metrics and update the Command Center application.

Greenplum Database calls the query metrics extension when a query is first submitted, when a query's status changes, and when a node in the query execution plan initializes, starts, or finishes. The query metrics extension sends metrics to the metrics collection agent running on each segment host. The extension also collects information about the locks queries hold so that you can see which queries hold locks that block other queries. The agent posts the metrics to the Greenplum Command Center rpc port.

The `metrics_collection` extension is included with Pivotal Greenplum Database. The extension is enabled by setting the `gp_enable_query_metrics` server configuration parameter to on and restarting the Greenplum Database cluster. The metrics collection agent is installed on each host when you install Greenplum Command Center. The Command Center application monitors the agent and restarts it if needed.

Command Center Workload Management

Workloads set concurrency, memory, and CPU resource limits for database transactions they manage. A Greenplum Command Center workload corresponds to a Greenplum Database resource group, but adds additional capabilities that are not available with resource groups.

Command Center allows administrators greater flexibility in assigning transactions to workloads. Every Greenplum Database role is assigned to a single resource group and, by default, transactions are managed by the role's resource group. With Command Center workload management, administrators can define criteria to assign transactions to workloads based on attributes other than the role submitting the transaction. Currently, assignment criteria can evaluate query tags and roles in combination with query tags.

A *query tag* is a key-value pair defined in the `gpcc.query_tags` parameter of a database session. The parameter has the format `<tag1>=<value1>;<tag2>=<value2>`, where tags and values are user-defined values. For example, if you want to run ETL operations in a workload named "etl", you could define a tag named "xact-type" and set it to "etl": `xact-type=etl`. The `gpcc.query_tags` parameter can be set as a connection parameter on Greenplum Database clients that allow it, or with a `SET` command inside the session after the connection has been established, for example `SET gpcc.query_tags='xact-type=etl'`.

The `gp_wlm` extension in Pivotal Greenplum Database provides support for Command Center workloads. Initially, Greenplum Database uses resource queues to manage resources. Using Command Center workloads requires enabling resource groups in Greenplum Database. Resource groups are based on the Linux control groups (cgroups) service, which must first be enabled in the operating system.

Installing Greenplum Command Center

Perform these tasks to install Greenplum Command Center on your Greenplum Database system:

- [Installing Greenplum Command Center](#)
- [Setting the Command Center Environment](#)

Additional topics:

- [Upgrading Greenplum Command Center](#)
- [Uninstalling Greenplum Command Center](#)

Installing Pivotal Greenplum Command Center

The Pivotal Greenplum Command Center installation utility installs the Command Center files on all hosts in the Greenplum Database cluster.

Run the Greenplum Command Center installer on the Greenplum Database master host. The installer installs the Command Center software on every host in your Greenplum Database cluster. It retrieves the list of hosts in your Greenplum Database cluster from the `gp_segment_configuration` system table.

After you have run the installer you can start Greenplum Command Center on the master host (recommended) or on the standby master host.

Prerequisites

Before installing Greenplum Command Center, ensure the following requirements are met:

- Greenplum Database must be installed and running. See the Pivotal Greenplum Command Center release notes for compatible Greenplum Database versions.
- The Greenplum Database `MASTER_DATA_DIRECTORY` environment variable must be set.
- The directory where Greenplum Command Center will be installed, `/usr/local/` by default, must be writable by the `gpadmin` user on all Greenplum Database hosts. See [Selecting and Preparing an Installation Directory for Command Center](#).
- Port 28080 (default) must be open to TCP connections from Web clients to the HTTP server on the master and standby master hosts. Greenplum Command Center web browser clients connect to this port to access the Command Center Console. Browser connections use HTTP/HTTPS and WebSocket (WS)/Secure WebSocket (WSS) protocols. A different port number can be specified when Command Center is installed. To access the Command Center web server through a proxy, the proxy must have WebSocket support.
- Port 8899 must be open on all hosts in the Greenplum Database cluster for TCP

connections. This is an RPC port, used by the metrics collection agents on the segment hosts to send metrics to the backend.

- Command Center requires the Apache Portable Runtime Utility library. The library is no longer included in the Greenplum Database lib directory, but it is installed as a dependency if you install the Greenplum Database distribution with `yum` or `apt`. Run the command `yum install apr-util` or `apt install libapr1` if you need to install the `apr-util` library.
- If you want Command Center server to support SSL/TLS encryption for browser connections, you need a combined SSL certificate file containing the server certificate and private key. See [SSL/TLS Encryption](#) for more information.

Selecting and Preparing an Installation Directory for Command Center

The Command Center installation directory (default `/usr/local`) must exist and be writable on every host in the Greenplum Database cluster by the `gpadmin` user. The Command Center installer creates a directory named `greenplum-cc-web-<version>` in the installation directory on every host. When Command Center installation is complete the `greenplum-cc-web-<version>` directory and all of its contents must be owned by the `gpadmin` user.

In a standard Linux system, the `/usr/local` directory is owned by `root` and is only writable by `root`. If you choose the default installation directory or another directory where `gpadmin` does not have write permission, you must make the directory writable by `gpadmin` on each host in the cluster.

You can use the `gpssh` utility to set permissions on all segment hosts at once if the `gpadmin` role has `sudo` access. For example:

```
$ source /usr/local/greenplum-db-<version>/greenplum_path.sh
$ gpssh -f <hostfile> 'sudo chmod 777 /usr/local'
```

After the Command Center installation is complete, you can restore the previous permissions on the installation directory.

See the *Pivotal Greenplum Database Installation Guide* for information about setting up passwordless SSH.

Install the Greenplum Command Center Software

Run the Greenplum Command Center installer on the Greenplum Database master host as the `gpadmin` user. The installer copies the software to all other hosts in the cluster.

1. Download the Greenplum Command Center distribution file for your Greenplum Database version from [Pivotal Network](#) and copy it to the `gpadmin` user's home directory on the master host.
2. Extract the installer from the zip file.

```
$ unzip greenplum-cc-web-<version>-LINUX-x86_64.zip
```

Extracting the installer creates a `greenplum-cc-web-<version>` directory containing the `gpccinstall-<version>` installation utility.

There are four ways to run the Greenplum Command Center installer:

- **Interactive** – the installer prompts you for the installation parameters.
- **Scripted** – you run the installer with a configuration file containing installation parameters.
- **Upgrade** – the installer uses the installation parameters from the current Command Center installation.
- **Auto** – the installer uses default installation parameters.

Interactive Installation

1. Source the `greenplum_path.sh` script in the Greenplum Database installation directory to ensure the `GPHOME` environment variable is set.

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

2. Run the Greenplum Command Center installer.

```
$ cd greenplum-cc-web-<version>
$ ./gpccinstall-<version>
```

You can add the following options to the `gpccinstall` command-line.

- ◊ The `-w` option instructs the installer to prompt for the gpmon database user's password.
 - If the gpmon user does not yet exist, the installer creates it using the password you enter at the prompt. The password is not saved anywhere, so be sure to remember the password you enter.
 - If the gpmon user already exists, the installer uses the password you enter at the prompt for the gpmon user. It ignores any password set in the `PGPASSWORD` environment variable, or in the `.pgpass` file in the `gpadmin` user's home directory (or in the file specified in the `PGPASSFILE` environment variable).
 - If the gpmon user already exists, but you do not specify the `-w` option, the installer uses the password set in the `PGPASSWORD` environment variable or in the `.pgpass` file.
 - If the gpmon user does not yet exist and you do not specify `-w` (or if you specify the `-w` option but enter no password at the prompt) the installer creates the gpmon user with a default password. See the `.pgpass` file to find the password.
 - ◊ The `--ssh_path` option allows you to specify the full path to a custom SSH program. If you do not supply this option, the installer uses the `ssh` command on the path.
Example: `bash $./gpccinstall-<version> --ssh_path /usr/local/bin/ssh -w`
 - ◊ The `-krbenable` option includes prompts to configure Command Center Kerberos authentication. The Kerberos prompts are omitted from the installation dialog by default.
3. Read through the license agreement and enter `y` to accept.

4. **Where would you like to install Greenplum Command Center? (Default=/usr/local)**

Press Enter to accept the default or enter the desired path. The directory must exist on all hosts in the Greenplum Database cluster and must be writable by the gpadm user.

5. **What would you like to name this installation of Greenplum Command Center? (Default=gpsc)**

Enter a name to display on Command Center web pages to identify this Greenplum Command Center installation.

6. **What port would you like the gpcc webserver to use? (Default=28080)**

The default Command Center listen port is 28080. Press Enter to accept the default or enter another port number.

7. (Optional). These prompts appear only if you specify `-krbenable` on the `gpccinstall` command line.

Enter webserver name for this instance: (Default=<gpcc-host>)

The Kerberos keytab file must contain a principal for the Command Center web server. The web server principal name has the the format `HTTP/<gpcc-host>@<realm>`, where `<gpcc-host>` is the host name clients use in URLs when connecting to the Command Center web server.

Enter the name of the Kerberos service: (Default=postgres)

The default Kerberos service name for Greenplum Database is `postgres`.

Choose Kerberos mode (1.normal/2.strict/3.gpmon_only): (Default=1)

Greenplum Command Center supports three different Kerberos authentication schemes.

1 - normal mode (default) – The Command Center Kerberos keytab file contains the Command Center principal and may contain principals for Command Center users. If the principal in the client's connection request is in the Command Center's keytab file, Command Center uses the client's principal for database connections. Otherwise, Command Center uses the gpmon user for database connections.

2 - strict mode – Command Center has a Kerberos keytab file containing the Command Center service principal and a principal for every Command Center user. If the principal in the client's connection request is in the keytab file, the web server grants the client access and the web server connects to Greenplum Database using the client's principal name. If the principal is not in the keytab file, the connection request fails.

3 - gpmon_only mode – Command Center uses the gpmon database role for all Greenplum Database connections. No client principals are needed in the Command Center's keytab file.

Enter path to the keytab file

Enter the path to the keytab file containing the Kerberos principal for the Command Center web server and, optionally, Command Center user principals.

What is the URL of the Command Center web server?

The Kerberos keytab file must contain a principal for the Command Center web server. The principal name is in the format `HTTP/<gpcc-host>@<realm>`, where `<gpcc-host>` is the host name clients use in URLs when connecting to the Command Center web server.

8. Would you like to enable SSL?

Enter `y` if you want to enable SSL/TLS (HTTPS) encryption for client connections to the Command Center web server. The installation utility prompts for the location of the combined SSL certificate file.

Enter the full path of the certificate file, including file name

Enter the path to the combined SSL certificate file installed on the Command Center host. This file contains a certificate and a private key for the web server. The file must be readable by the `gpadm` user. See [SSL/TLS Encryption](#) for information about creating this file and installing it on your server.

9. Choose a display language (Default=English)

1. English
2. Chinese
3. Korean
4. Russian
5. Japanese

Enter a number to choose a language for the Command Center user interface.

The installer saves a log of the installation session in the current directory in a file named `gpccinstall.<timestamp>.log`.

Install With a Configuration File

You can provide a configuration file to the Greenplum Command Center installer to perform a non-interactive Command Center installation. Note that you must still view and accept the license agreement.

```
$ cd greenplum-cc-web-<version>
$ ./gpccinstall-<version> -c <config-file>
```

The following table contains the names of the parameters corresponding to the interactive installation prompts and their default values. Define parameters in a configuration file for any parameters that have no default value or to override default values.

Installer Prompt	Default	Parameter
Where would you like to install Greenplum Command Center?	<code>/usr/local</code>	<code>path</code>
What would you like to name this installation of Greenplum Command Center?	<code>gpcc</code>	<code>display_name</code>
What port would you like the gpcc webserver to use?	<code>28080</code>	<code>web_port</code>

| Would you like to enable Kerberos? | `false` | `enable_kerberos` | | Choose Kerberos mode (1.normal/2.strict/3.gpmon_only): | `1` | `krb_mode` | | Please provide the path to the keytab file: | | keytab | | What is the name of the GPDB Kerberos service? | `postgres` | `krb_service_name` | | What is the URL of the Command Center web server? | `gpcc` | `webserver_url` | | Would you like to enable

SSL? | false | enable_ssl | | Please provide the file path for the SSL certificate: |
 /etc/ssl/certs/cert.pem | ssl_cert_file | | Please choose a display language
 (1.English/2.Chinese/3.Korean/4.Russian/5.Japanese) | 1 | language |

If the `enable_kerberos` parameter is true, the `keytab`, `webserver_url`, `krb_mode`, and `krb_service_name` parameters must also be set.

If the `enable_ssl` parameter is true, the `ssl_cert_file` parameter is required.

The following installation configuration file example sets all parameters to their default values.

```
path = /usr/local
# Set the display_name param to the string to display in the GPCC UI.
# The default is "gpcc"
# display_name = gpcc

master_port = 5432
web_port = 28080
rpc_port = 8899
enable_ssl = false
# Uncomment and set the ssl_cert_file if you set enable_ssl to true.
# ssl_cert_file = /etc/certs/mycert
enable_kerberos = false
# Uncomment and set the following parameters if you set enable_kerberos to true.
# webserver_url = <webserver_service_url>
# krb_mode = 1
# keytab = <path_to_keytab>
# krb_service_name = postgres
# User interface language: 1=English, 2=Chinese, 3=Korean, 4=Russian, 5=Japanese
language = 1
```

Non-Interactive Installation with Defaults

The non-interactive installation is useful when installing Command Center in a cloud environment.

1. Source the `greenplum_path.sh` script in the Greenplum Database installation directory to ensure the `GPHOME` environment variable is set.

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

2. Run the Greenplum Command Center installer with the `-auto` option.

```
$ cd greenplum-cc-web-<version>
$ ./gpccinstall-<version> -auto
```

Upgrade

Running `gpccinstall` with the `-u` option installs a Greenplum Command Center release using the configuration parameters from the current Command Center installation. You can install a new Command Center release, or reinstall the current release. This option is useful after you have added new hosts to the Greenplum Database cluster or replaced failed hosts.

The configuration parameters are read from the `$GPCC_HOME/conf/app.conf` file.

1. Source the `greenplum_path.sh` script in the Greenplum Database installation directory to

ensure the `GPHOME` environment variable is set.

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

2. Source the `gpcc_path.sh` script in the Greenplum Command Center installation directory.

```
$ source /usr/local/greenplum-cc-web-<version>/gpcc_path.sh
```

3. Run the Greenplum Command Center installer with the `-u` option.

```
$ cd greenplum-cc-web-<version>
$ ./gpccinstall-<version> -u
```

Prepare the Standby Master Host

After the Command Center software is installed, you start the Command Center web server and backend on the master host or the standby master host. Running Command Center on the master host is recommended, but preparing the standby host ensures the standby can run Command Center if the master fails.

1. Copy the `.pgpass` file from the master host to the standby master host. Run these commands from the master:

```
$ ssh gpadmin@<standby_host>
$ scp gpadmin@<master_host>:~/.pgpass ~
$ chmod 600 ~/.pgpass
```

NOTE There are two alternative methods you can use to supply the `gpmon` user's password so that you do not have to put the `.pgpass` file on the host. Command Center only requires the `gpmon` password when you run `gpcc start`, `gpcc stop`, or `gpcc status`.

1. Set the `PGPASSWORD` environment variable before you run `gpcc` commands. Example:

```
$ PGPASSWORD=changeme gpcc status
```

2. Add the `-W` option to `gpcc` commands to have the command prompt you for the password. Example:

```
$ gpcc start -W
```

2. If the Greenplum Command Center web server is to support TLS/SSL, a server certificate `.pem` file must be obtained and installed on the standby host in a location readable by the `gpadmin` user. The default path is `/etc/ssl/certs/cert.pem`.
3. If Greenplum Command Center is to support Kerberos authentication, Greenplum Database must have Kerberos authentication set up and the required principals and keytabs must be installed on the standby host. See [Enabling Authentication with Kerberos](#) for Kerberos setup instructions. You can install Command Center without Kerberos authentication initially and then enable Kerberos later by running the `gpcc krbenable` command.

Next Steps

- [Setting the Greenplum Command Center Environment](#)
- [Starting and Stopping Greenplum Command Center](#)
- [Connecting to Greenplum Command Center](#)

Upgrading Greenplum Command Center

To upgrade Greenplum Command Center, you install the new Command Center software release, stop the old version, and start the new version. You can then remove the older Command Center release from your Greenplum Database hosts.

Upgrading From Greenplum Command Center 3.x to 4.x

The Greenplum Command Center architecture changed between Command Center 3.x and 4.x.

With Command Center 3.x, you installed the Greenplum Command Center software one time on the Command Center host. You then created a Command Center instance for each Greenplum Database cluster you monitored with Command Center.

Command Center 4.x does not have instances; you install the Command Center software on the master or standby master of the Greenplum Database cluster you want to monitor with Command Center. The installer copies the software to every host in the Greenplum Database cluster. To monitor additional Greenplum Database clusters you must install the Command Center software again, on a different master or standby host. Monitoring multiple Greenplum Database clusters running on the same hardware cluster is not supported.

To upgrade to a new release of Greenplum Command Center 4.x:

1. Download and install the new Command Center release by following the instructions in [Installing Greenplum Command Center](#).
2. Stop the current Command Center release.

Command Center 3.x:

```
$ gpccmdr --stop <instance_name>
```

Command Center 4.x:

```
$ gpcc stop
```

3. Source the `gpcc_path.sh` script in the new Command Center installation directory.

```
$ source /usr/local/greenplum-cc-web-<version>/gpcc_path.sh
```

Note: Also update the source command in your shell start-up script, for example `~/.bashrc.sh` or `~/.bash_profile.sh`.

4. Start the new Command Center release.

```
$ gpcc start
```

5. Uninstall the older Command Center release. See “Uninstalling Greenplum Command

Center” in the Greenplum Command Center documentation for the release you are uninstalling.

Uninstalling Greenplum Command Center

To uninstall Greenplum Command Center, you must stop both the Command Center Console and disable the data collection agents. Optionally, you may also remove any data associated with Greenplum Command Center by removing your Command Center Console installation and the gpperfmon database.

1. Stop Command Center Console if it is currently running. For example:

```
$ gpcc --stop
```

2. Remove the Command Center installation directory from all hosts. For example:

```
$ rm -rf /usr/local/greenplum-cc-web-version
```

3. Disable the data collection agents.
 1. Log in to the master host as the Greenplum administrative user (gpadmin):

```
$ su - gpadmin
```

2. Disable the data collection agents by setting the `gp_enable_gpperfmon` server configuration parameter off:

```
$ gpconfig -c gp_enable_gpperfmon -v off
```

3. Remove or comment out the gpmon entries in `pg_hba.conf`. For example:

```
#local      gpperfmon      gpmon      md5
#host       gpperfmon      gpmon      0.0.0.0/0      md5
```

4. Drop the Command Center superuser role from the database. For example:

```
$ psql template1 -c 'DROP ROLE gpmon;'
```

5. Restart Greenplum Database:

```
$ gpstop -r
```

6. Clean up any uncommitted Command Center data and log files that reside on the master file system:

```
$ rm -rf $MASTER_DATA_DIRECTORY/gpperfmon/data/*
$ rm -rf $MASTER_DATA_DIRECTORY/gpperfmon/logs/*
```

7. If you do not want to keep your historical Command Center data, drop the gpperfmon database:

```
$ dropdb gpperfmon
```

Setting the Greenplum Command Center Environment

To enable the `gpadmin` user to execute Command Center utilities such as `gpcc` at the command line, source the `gpcc_path.sh` file in the Greenplum Command Center installation directory. For example:

```
$ source /usr/local/greenplum-cc-web-<version>/gpcc_path.sh
```

The `gpcc_path.sh` script sets the `GPCC_HOME` environment variable to the Command Center installation directory and adds the `$GPCC_HOME/bin` directory to the path.

To automatically source the `gpcc_path.sh` each time you log in, add the above source command to your start-up script, for example `~/.bashrc` or `~/.bash_profile`.

Connecting to the Greenplum Command Center Console

Sign in to the Command Center Console with a name and password. If the Guest Access to Query Monitor feature is enabled, you can sign in anonymously to see just the **Query Monitor** view.

Open the Command Center Console in a supported browser using the host name and port configured for the Command Center web server. For example, to open a secure Command Center connection on a host named `smdw` at port 28080, enter this URL into your browser:

```
https://smdw:28080
```

Welcome to GPCC 4.4

What's new

- Unique URLs you can bookmark and share
- Interactive History view
- Administrative Permissions & Access controls

What's next

Let us know what's important to you in our [GPCC Roadmap Prioritization Survey](#)

We want your feedback

For GPCC and WLM
gpcc-feedback@pivotal.io
 For all other Greenplum topics
gpdb-feedback@pivotal.io

Sign In

Server: **gpcc4.4**

Username

Password

Remember my username

SIGN IN

View Query Monitor
without signing in

© 2018 Pivotal Software, Inc. All Rights Reserved. [Privacy Policy](#) - [Terms of Service](#)

- If the **View Query Monitor** link is present, you can click it to view the **Query Monitor** page without signing in. This takes you immediately to the **Query Monitor** view. To access additional Command Center features, click **Sign In** on the **Query Monitor** view and sign in with a valid Command Center user name and password. If the link is not present on the sign-in page, a Command Center administrator has disabled the anonymous query monitor feature.
- To sign in as a Command Center user, enter the user name and password of a Greenplum role that has been configured to allow authentication to Greenplum Command Center, then click **Sign In**. This opens the **Dashboard** page of the Command Center Console, which provides a graphical system snapshot and a summary view of active queries. See the [Dashboard](#) for information about the Dashboard view.

Note to Chrome Browser Users

If you install a new version of Greenplum Command Center using the same port number as the previous version, and you use the Chrome web browser, you may be unable to view real-time queries until after you clear the browser's cache. Follow these steps.

1. Choose **Settings** from the Chrome menu.
2. Scroll to the bottom and click **Advanced**.
3. Under **Privacy and security**, click **Clear browsing data**.
4. Click the **Basic** tab and select **Cached images and files**. You do not have to clear **Browsing history** or **Cookies and other site data**.
5. Click **CLEAR DATA** and then log in to Command Center.

Administering the Command Center Web Server

The `gpccws` web server binary and web application files are installed in the `bin` directory of your Greenplum Command Center installation.

Starting and Stopping the Web Server

Starting the Command Center Web Server runs the `gpccws` web server, starts the metrics collection agents on the segment servers, and starts a listener on the Command Center rpc port.

You can run the `gpcc` command as the `gpadmin` user on the master host (recommended) or on the standby host. Starting Command Center on the standby host is not recommended because it can cause heavy network traffic between the Command Center agent on the master host and the backend on the standby host.

To ensure the `gpcc` command is on your path, source the `gpcc_path.sh` file in the Command Center installation directory or add it to the startup script for your command shell. See [Setting the Greenplum Command Center Environment](#) for instructions. The `MASTER_DATA_DIRECTORY` environment variable must be set to the location of the Greenplum Database master data directory.

NOTE

The `gpcc` command uses the `gpmon` role to connect to Greenplum Database. It looks for the `gpmon` password in the `PGPASSWORD` environment variable or in the `.pgpass` file in the `gpadmin` user's home directory. You can instead append the `-w` flag to the `gpcc` commands below to have `gpcc` prompt you to enter the password.

To start Greenplum Command Center

Log on to the master host or the standby host.

To log on to the standby from the master host:

```
$ ssh <standby-host>
```

Source the Command Center environmental script.

```
$ source /usr/local/greenplum-cc-<version>/gpcc_path.sh
```

Start the Command Center web server and the metrics collection agents.

```
$ gpcc start
Starting the gpcc agents and webserver...
2019/05/07 01:51:03 Agent successfully started on 5/5 hosts
2019/05/07 01:51:03 View Greenplum Command Center at http://mdw:28090
```

To stop Greenplum Command Center

```
$ gpcc stop
2019/05/07 01:51:55 GPCC webserver and metrics collection agents have been stopped. Use
gpcc start to start them again
```

To check the Greenplum Command Center status

```
$ gpcc status
2019/05/07 01:50:13 GPCC webserver: running
2019/05/07 01:50:14 GPCC agents: 5/5 agents running
```

See the [gpcc](#) reference page for full syntax for the `gpcc` command.

Configuring the Command Center Web Server

The web server configuration file is stored in `$GPCC_HOME/conf/app.conf`. The parameters in this configuration file are set when you install Greenplum Command Center. The installer copies the Command Center installation directory, including this configuration file, to every Greenplum Database host.

See the *Web Server Parameters* section of [Configuration File Reference](#) for a description of the parameters in this file.

You can see a summary of the current configuration using the `gpcc --settings` command.

```
$ gpcc --settings
Install path:          /usr/local
Display Name:         gpcc
GPCC port:            28080
Kerberos:             enabled
Kerberos webserver name: mdw
Kerberos gpdb name:   gpadmin
Kerberos mode:        gpmon_only
Kerberos path:        /home/gpadmin/gpcc-kerberos.keytab
SSL:                  disabled
```

If you modify the file on one host you should copy it to every other host. Be sure to restart the web server after you change the configuration. Rather than modifying the configuration directly, you can just stop Command Center and re-run the `gpccinstall-<version>` installation command. This ensures the configuration is consistent on all hosts.

You can use the `gpcc krbenable` command to add Kerberos authentication to the Command Center configuration. See [Enabling Authentication with Kerberos](#) for details about setting up Kerberos on the Command Center host. The `gpcc krbenable` command prompts for the Kerberos principal names and artifacts and updates the configuration.

The `gpcc krbdisable` command removes Kerberos parameters from the Command Center configuration.

Viewing and Maintaining Web Server Log Files

Web server access and error log messages are written to `$GPCC_HOME/logs/gpccws.log`.

If you experience errors viewing the Greenplum Command Center Console, refer to this file for more information.

To prevent the web server log from growing to excessive size, you can set up log file rotation using `logrotate` or `crondlog`.

Administering Command Center Agents

The Command Center metrics collection agent, `ccagent`, runs on segment hosts and receives real-time metrics emitted by the metrics collection database extension. Each segment host has one `ccagent` process. The metrics collection extension connects to `ccagent` using Unix Domain Sockets (UDS) to transfer metrics from Greenplum Database. Starting Greenplum Command Center with the `gpcc start` command starts the Command Center agent on each segment host. Stopping Command Center with `gpcc stop` ends these processes. The Command Center backend monitors these agents and restarts them when necessary.

This topic describes basic agent administration tasks, including enabling agents after adding hosts to the cluster and viewing the agent log files.

About the gpperfmon Data Collection Agents

The Greenplum Database `gpperfmon_install` utility enables the `gpmmmon` and `gpsmon` data collection agents. Greenplum Command Center no longer requires the history data these agents collect. You can run the gpperfmon data collection agents and the Command Center metrics collection agents in parallel, but unless you need the data the gpperfmon agents collect for some other purpose, you can improve the Greenplum Database system performance by disabling the gpperfmon agents.

To see if the gpperfmon agents are enabled, check the `gp_enable_gpperfmon` server parameter with the following command.

```
$ gpconfig -s gp_enable_gpperfmon
```

If the `gp_enable_gpperfmon` parameter is on, disable the agents by setting the parameter to off and restarting Greenplum Database.

```
$ gpconfig -c gp_enable_perform -v off
$ gpstop -ar
```

For more information about the gpperfmon data collection agents see `gpperfmon_install` in the *Greenplum Database Utility Reference*.

Configuring gpmon Role Logging Options

The metrics collection agent `ccagent` runs queries on Greenplum Database on behalf of Command Center to perform activities such as retrieving information to display in the Command Center UI, saving state in the gpperfmon and postgres databases, inserting alert event records, and harvesting query history for the gpmetrics history tables. The agent runs these queries using the gpmon database role.

If you do not want all of this query activity to be logged in the `pg_log` log file, you can set logging options for the `gpmon` role using the `ALTER ROLE` command. For example, these commands set the `gpmon` role to log only DDL statements (`CREATE`, `ALTER`, `DROP`) and to log only fatal and panic errors.

```
ALTER ROLE gpmon SET log_statement TO DDL;  
ALTER ROLE gpmon SET log_min_messages to FATAL;
```

See the `log_statement` and `log_min_messages` server configuration parameters in the Greenplum Database documentation for logging options.

Adding and Removing Hosts

When you add or replace Greenplum Database hosts, you must reinstall the Greenplum Command Center software to ensure the software is installed on the new hosts. It is not necessary to uninstall Command Center before reinstalling.

Viewing and Maintaining Agent Log Files

Log messages for the Command Center metrics collector agents are saved in the `logs` directory of the Command Center installation directory on the master and each segment host.

The current log file is named `agent.log`. When a new log is started, the current log is renamed to `agent.<timestamp>.log` and a new `agent.log` file is opened. You should remove old log files regularly to recover space.

Administering the gpperfmon Database

Data collected by gpmmmon and gpsmon agents is stored in a dedicated database called gpperfmon. This database requires the typical database maintenance tasks, such as clean up of old historical data and periodic `ANALYZE`.

See the [gpperfmon Database Reference](#) section for a reference of the tables and views in the gpperfmon database.

Connecting to the gpperfmon Database

Database administrators can connect directly to the gpperfmon database using any Greenplum Database-compatible client program (such as `psql`). For example:

```
$ psql -d gpperfmon -h master_host -p 5432 -U gpadmin
```

Backing Up and Restoring the gpperfmon Database

The history tables of the gpperfmon database can be backed up and restored using the Greenplum Database backup and restore utilities. See the *Greenplum Database Utility Guide* for more information.

Maintaining the Historical Data Tables

All of the `*_history` tables stored in the gpperfmon database are partitioned into monthly partitions. A January 2010 partition is created at installation time as a template partition. It can be deleted once some current partitions are created. The Command Center agents automatically create new partitions in two month increments as needed. Administrators must periodically drop partitions for the months that are no longer needed in order to maintain the size of the database.

See the *Greenplum Database Administrator Guide* for more information on dropping partitions of a partitioned table.

Settings

The **Admin > Settings** view enables Command Center administrators to configure settings for Command Center features.

History settings

Turn on **Enable GPCC history data collection** to save query history, host metrics, and disk usage metrics to tables in the gpperfmon database gmetrics schema. This option defaults to on for a Greenplum Database 6 installation, but the Greenplum Database metrics collector extension is inactive until Command Center is started.

Enter a number of seconds to set the minimum runtime for a query to be saved in history. The default is to save all queries. Set this threshold to prevent Command Center from filling history with trivial queries.

The screenshot shows the 'Settings' page for 'Pivotal Greenplum Command Center'. The 'History settings' section is active, showing a toggle for 'Enable GPCC history data collection' which is turned on. Below this, there is a checkbox for 'Skip history collection for queries shorter than' followed by an input field containing '10' and the word 'seconds'. A note explains that queries shorter than the specified time will not be collected in the gppcc_queries_history table to prevent resource overhead for large transactional workloads. The page also includes a sidebar with navigation options like Dashboard, Query Monitor, Host Metrics, Cluster Metrics, History, System, Admin, Permissions, Authentication, Workload Mgmt, Alerts, and Settings. The top navigation bar shows 'server: gpcc' and 'Welcome, gpmon'.

When you enable GPCC history data collection:

- Command Center saves query and metrics history in the gmetrics schema tables in the gpperfmon database. Only queries that execute for at least the number of seconds you specify are saved. Query plan node history is only saved for queries that run for at least 10 seconds, or the number of seconds you specify, if greater than 10.

See [gpmetrics Schema Reference](#) for information about the gpmetrics history tables.

When you disable GPCC history data collection:

- Command Center query history views use history data collected by the Greenplum Database gpperfmon agents.

For best performance, set **Enable GPCC history data collection** to on and disable gpperfmon data

collection. To disable the gpperfmon agents, set the Greenplum Database `gp_enable_gpperfmon` configuration parameter to off and restart Greenplum Database:

```
$ gpconfig -c gp_enable_gpperfmon -v 'off'  
$ gpstop -ar
```

gpcc

Manages the Greenplum Command Center web service and metrics collection agents.

```
gpcc <action> [-W]
gpcc [--version | -v ]
gpcc [--help | -h]
gpcc [--settings]
```

Actions

start

Starts the Command Center web service and metrics collection agents. Add the `-w` flag to force a prompt for the gpmon user password.

stop

Stops the Command Center web service and metrics collection agents. Add the `-w` flag to force a prompt for the gpmon user password.

status

Displays the status, either `Running` or `Stopped`, of the web server and metrics collection agents. Add the `-w` flag to force a prompt for the gpmon user password.

krbenable

Enables Kerberos authentication for Command Center.

Use the `gpcc krbenable` command to set up Kerberos authentication for Command Center users if Command Center was initially installed without enabling Kerberos. When you run `gpcc krbenable`, `gpcc` prompts for:

- the web server name
- the name of the Greenplum Database Kerberos service
- the Command Center Kerberos authentication mode
- the path to the keytab file on the Command Center host.

Before you run `gpcc krbenable`, see [Enabling Authentication with Kerberos](#) to check prerequisites and for help preparing the Command Center host to allow Kerberos authentication.

krbdisable

Disables Kerberos authentication for Command Center.

help

Displays syntax and help text for the `gpcc` command.

Options

--settings

Displays the current values of the Command Center configuration parameters. See [Setup Configuration File](#) for a list of the configuration parameters.

--version or **-v**

Displays the Greenplum Command Center version.

-W <password>

The optional `-w` option specifies the password for the `gpmon` user. The `gpcc` command normally gets the password from the `$PGPASSWORD` environment variable or the `.pgpass` file in the `gpadmin` user's home directory. If the password is not available with either of these methods, the `-w` option must be included to specify the password whenever you run `gpcc`.

Description

The Greenplum Database `MASTER_DATA_DIRECTORY` environment variable must be set when you run the `gpcc` command. This environment variable is usually set in the `gpadmin` user's shell initialization script (`/home/gpadmin/.bashrc`, for example). If `MASTER_DATA_DIRECTORY` is not set when you run `gpcc start`, `gpcc` prints an error message and exists.

Once started, the Command Center backend monitors the metrics agents with a heartbeat. If a failed agent is detected, the backend spawns a new agent process.

Examples

Start Command Center and the metrics agents, prompting for the `gpmon` password.

```
$ gpcc start -W
Password for GPDB user gpmon:
Starting the gpcc agents and webserver...
2018/03/22 17:51:51 Agent successfully started on 7/8 hosts
2018/03/22 17:51:51 View Greenplum Command Center at http://smdw:28080
```

gpmetrics Schema Reference

Greenplum Command Center creates the `gpmetrics` schema in the Greenplum Database `gpperfmon` to save alert rules and logs, and historical metrics collected by the Greenplum Database metrics collection system. The `gpperfmon` schema contains the following tables:

- `gpcc_alert_rule` — saves alert rules configured on the Command Center **Admin> Alerts** page.
- `gpcc_alert_log` — records an event when an alert rule is triggered.
- `gpcc_database_history` — saves summary query activity information.
- `gpcc_disk_history` — saves disk usage statistics for each Greenplum Database host file system.
- `gpcc_plannode_history` — saves plan node execution statistics for completed queries.
- `gpcc_queries_history` table — saves execution metrics for completed Greenplum Database queries.
- `gpcc_system_history` table — saves system metrics sampled from Greenplum Database segments hosts.

If you set the schema search path to include the `gpmetrics` schema, you do not have to qualify table names with the `gpmetrics` schema name. To set the default search path for the `gpperfmon` database enter this SQL command.

```
=# ALTER DATABASE gpperfmon SET search_path TO public, gpmetrics;
```

You must exit the current session and start a new session for the new search path to take effect.

Alert Tables

Command Center uses the `gpcc_alert_rule` and `gpcc_alert_log` tables to store the alert rules you set up in the user interface and to log messages when the alert rules are triggered.

gpcc_alert_rule Table

The `gpcc_alert_rule` table records the alert rules configured in the Command Center user interface. It has the columns shown in the following table.

Column	Type	Description
<code>rule_id</code>	integer	Unique id for the rule.
<code>rule_type</code>	integer	Reserved for future use.

Column	Type	Description
rule_description	character varying(512)	Text of the rule.
rule_config	json	JSON string containing parameters for user-specified values.
ctime	timestamp(0) without time zone	Time the rule was created.
etime	timestamp(0) without time zone	Time the rule became inactive, or null if rule is active.

The `gpcc_alert_rule` table keeps a history of alert rule configurations. When a rule becomes active, a new row is inserted and the `ctime` timestamp column is set to the current time; the `etime` timestamp is null, indicating that the rule is still active. When a rule is either disabled or superseded by a new rule, the `etime` timestamp column is set to the current time. Thus, the set of currently active rules is all rows where the `etime` column is null. A row that has timestamps in both `ctime` and `etime` columns is an historical record of the period of time during which the rule was active.

The `rule_id` column, a unique integer, is the distribution key for the table and is used to identify a single alert rule configuration. This column can be joined with the `rule_id` column in the `gpcc_alert_log` table to identify the rule that triggered each recorded alert event.

The `rule_description` column contains a string that describes the event that matches the rule. It is the text displayed in the Command Center UI for the rule, with user-specified values inserted.

The `rule_config` column contains a JSON string with parameters for the values entered for each of the rule's fields in the Command Center UI.

gpcc_alert_log Table

The `gpcc_alert_log` table has the columns shown in the following table.

Column	Type	Description
id	integer	Unique ID for the alert.
rule_id	integer	The ID of the rule that triggered this alert.
transaction_time	timestamp(0) without time zone	Time the alert was raised.
content	json	Contains parameters specifying values that triggered the alert.

The `gpcc_alert_log` table is an append-only, column-oriented table, partitioned by month on the `transaction_time` column. Command Center creates new partitions as needed and removes partitions over 12 months old.

A row is added to the `gpcc_alert_log` table whenever an alert rule is matched.

The `id` column, a unique integer, is the distribution key for the table.

The `transaction_time` column is set to the current time when a row is created.

The `rule_id` column can be joined with the `rule_id` column in the `gpcc_alert_rule` table to access details of the rule that triggered the alert.

The `content` column contains a JSON string with parameters specifying details about the event that triggered the alert. The JSON parameters vary with the type of the alert.

Example Query

This query lists the ten most recent alerts, including the configuration of the rule that triggered each event.

```
gpperfmon=# SELECT l.transaction_time, l.rule_id, r.rule_description, l.content
FROM gpmetrics.gpcc_alert_log AS l, gpmetrics.gpcc_alert_rule AS r
WHERE l.rule_id = r.rule_id AND r.etime is NULL
ORDER BY l.transaction_time
LIMIT 10;
```

Greenplum Database Metrics History Tables

The `gpmetrics` query history saves information collected by the Greenplum Database metrics collection system and forwarded to Greenplum Command Center.

The distribution key for each table is a `ctime` timestamp column, which is the time when the row is added to the database. The tables are partitioned by year and month. Greenplum Command Center creates new partitions automatically as needed.

The history tables use append-optimized, column-oriented storage.

Command Center only saves queries with runtimes greater than the value of the `min_query_time` configuration parameter, found in the `$MASTER_DATA_DIRECTORY/gpmetrics/gpcc.conf` configuration file on the host executing Command Center. The default, 0, saves all queries in the history table. This parameter can be configured on the Command Center **Admin> Settings** page.

gpcc_database_history

The `gpcc_database_history` table saves summary query activity metrics collected by the Greenplum Database metrics collector. This data can be used to review the Greenplum Database query load over time.

Column	Type	Description
<code>ctime</code>	timestamp(0) without time zone	Time the record was created.
<code>queries_total</code>	integer	Total number of queries running and queued to run.
<code>queries_running</code>	integer	Number of queries currently running.
<code>queries_queued</code>	integer	Number of queries queued, but not yet running.
<code>queries_blocked</code>	integer	The number of queries started, but blocked by other transactions.

gpcc_disk_history

The `gpcc_disk_history` table saves historical disk usage statistics for each Greenplum Database segment host file system.

Column	Type	Description
<code>ctime</code>	timestamp(0) without time zone	Time the row was created.

Column	Type	Description
hostname	character varying(64)	Name of the segment host.
filesystem	text	Path to the segment's data directory.
total_bytes	bigint	Total size of the file system storage in bytes.
bytes_used	bigint	Number of storage bytes in use.
bytes_available	bigint	Number of storage bytes available.

gpcc_plannode_history

The `gpcc_plannode_history` table saves detailed metrics for each operation (node) in a completed query plan. Each row contains metrics for one operation that executed on one Greenplum Database segment. This information allows reconstructing the plan and execution metrics for a completed query.

Plan node history is only saved for queries that execute for 10 seconds or more. The `gpcc_queries_history`

Column	Type	Description
ctime	timestamp(0) without time zone	Time the row was created.
tmid	integer	A time identifier for the query. All records associated with a query will have the same <code>tmid</code> .
ssid	integer	Session id for the database connection. All records associated with the query will have the same <code>ssid</code> .
ccnt	integer	Command number within the session. All records associated with the query will have the same <code>ccnt</code> .
segid	integer	Id (<code>dbid</code>) of the segment for this plan node.
procid	integer	The postgres process ID for this plan node.
sliceid	integer	Id of the slice the operation belongs to. Operations that belong to the same slice execute in parallel.
nodeid	integer	The query plan node ID for this operation.
parent_nodeid	integer	The parent query plan node ID from the query plan.
node_type	character varying(64)	Name of the operation type.
tinit	timestamp(6) without time zone	Time the operation was initialized.
tstart	timestamp(6) without time zone	Time the operation started.
tfinish	timestamp(6) without time zone	Time the operation finished.
status	character varying(16)	Status of the operation: <code>initialize</code> , <code>executing</code> , or <code>finished</code> .
planrows	double precision	The number of output rows estimated for the operation.

Column	Type	Description
planwidth	integer	Width of output rows estimated for the operation.
start_cost	double precision	Number of page reads expected to produce first output row.
total_cost	double precision	Number of page reads expected to produce all output rows.
tuple_count	bigint	
ntuples	bigint	
nloops	bigint	
first_tuple	timestamp(6) without time zone	Time the operation produced the first output row.
rel_oid	oid	Object ID of the output rows produced by the operation.
relation_name	character varying(64)	Name of the table this operation processed, if any.
index_name	character varying(64)	Name of the index used by this operation, if any.
alias_name	character varying(64)	Alias for the relation declared in the SQL command.
node_seq	integer	
condition	text	Condition expression used for a filter or join operation.

gpcc_queries_history

The `gpcc_queries_history` table saves metrics for completed queries.

Column	Type	Description
ctime	timestamp(0) without time zone	Time the row was created.
tmid	integer	A time identifier for the query. All records associated with the query will have the same <code>tmid</code> .
ssid	integer	Session id for the database connection. All records associated with the query will have the same <code>ssid</code> .
ccnt	integer	Command number within this session. All records associated with the query will have the same <code>ccnt</code> .
username	character varying(64)	Role name that issued the query.
db	character varying(64)	Name of the database queried.
cost	double precision	Estimated cost to execute query, computed by the legacy planner or GPORCA.
tsubmit	timestamp(6) without time zone	Time the query was submitted.

Column	Type	Description
tstart	timestamp(6) without time zone	Time the query was started.
tfinish	timestamp(6) without time zone	Time the query finished.
status	character varying(16)	Status of the query – <code>abort</code> , <code>error</code> , or <code>done</code> .
rows_out	bigint	Number of rows returned by the query.
error_msg	text	Error message, if the query failed.
plan_gen	character varying(16)	<code>PLANNER</code> if query plan was generated by the legacy planner; <code>OPTIMIZER</code> if the plan was generated by GPORCA, the Pivotal query optimizer.
query_hash	character varying(64)	Hash code generated from the text of the query.
query_text	text	Complete text of the query. Some queries may be reformatted before storing in the history database.
application_name	character varying(64)	Name of the client application that established the database connection.
rsqname	character varying(64)	If the <code>gp_resource_manager</code> configuration parameter is <code>queue</code> , the name of the resource queue managing the query.
rsgname	character varying(64)	If the <code>gp_resource_manager</code> configuration parameter is <code>group</code> , the name of the resource group managing the query.
cpu_master	bigint	Total CPU usage for this query on the Greenplum Database master instance.
cpu_segs	bigint	Total CPU usage for this query across all segments, measured in milliseconds. This is the sum of the CPU usage values taken from all active primary segments in the database array.
cpu_master_percent	double precision	Average CPU percent usage on the master host during execution of this query.
cpu_segs_percent	double precision	Average CPU percent usage on the segment hosts during the execution of this query.
skew_cpu	double precision	Displays the amount of processing skew in the system for this query. Processing/CPU skew occurs when one segment performs a disproportionate amount of processing for a query. The skew is calculated from total CPU seconds used on all segments during the execution of the query.
skew_rows	double precision	Displays the amount of row skew in the system. Row skew occurs when one segment produces a disproportionate number of rows for a query.
memory	bigint	Total size of memory, in bytes, used by all segments to execute this query.
disk_read_bytes	bigint	Number of bytes read from disk.
disk_write_bytes	bigint	Number of bytes written to disk.

Column	Type	Description
spill_size	bigint	Total size of spill files used by all segments to execute this query.
rqpriority	character varying(16)	Priority setting for the resource queue managing this query. Blank if resource group management is enabled.

gpcc_system_history

The `gpcc_system_history` table saves historical system metrics for each Greenplum Database host, including the master, standby master, and segment hosts. The metrics include information about memory, CPU, disk, and network utilization.

Column	Type	Description
ctime	timestamp(0) without time zone	Time the row was created.
hostname	character varying(64)	Segment or master hostname associated with these system metrics.
mem_total	bigint	Total system memory in Bytes for this host.
mem_used	bigint	System memory used, in Bytes, for this host.
mem_actual_used	bigint	Actual memory used, in Bytes, for this host (not including the memory reserved for cache and buffers).
mem_actual_free	bigint	Free actual memory, in Bytes, for this host (not including the memory reserved for cache and buffers).
swap_total	bigint	Total swap space in Bytes for this host.
swap_used	bigint	Swap space used, in Bytes, for this host.
swap_page_in	bigint	Number of swap pages in.
swap_page_out	bigint	Number of swap pages out.
cpu_user	double precision	Percentage of time CPU processes execute in user mode.
cpu_sys	double precision	Percentage of time CPU processes execute in system (kernel) mode.
cpu_idle	double precision	Percentage idle CPU.
load0	double precision	CPU one-minute load average.
load1	double precision	CPU five-minute load average.
load2	double precision	CPU fifteen-minute load average.
quantum	integer	Interval between metrics collections.
disk_ro_rate	bigint	Disk read operations per second.
disk_wo_rate	bigint	Disk write operations per second.
disk_rb_rate	bigint	Bytes per second for disk read operations.
disk_wb_rate	bigint	Bytes per second for disk write operations.
net_rp_rate	bigint	Packets per second on the system network for read operations.

Column	Type	Description
net_wp_rate	bigint	Packets per second on the system network for write operations.
net_rb_rate	bigint	Bytes per second on the system network for read operations.
net_wb_rate	bigint	Bytes per second on the system network for write operations.

Configuration Files Reference

Configuration parameters for Greenplum Command Center are stored in the following files.

`$MASTER_DATA_DIRECTORY/gpperfmon/conf/gpperfmon.conf`, on Greenplum Database master host
Stores configuration parameters for the Greenplum Command Center agents.

`$GPCC_HOME/conf/app.conf`, on Command Center host.

Stores configuration parameters for the Command Center web application and web server.

`$MASTER_DATA_DIRECTORY/gpmetrics/gpcc.conf`

Stores configuration parameters for Command Center interface options and alert emails.

`$MASTER_DATA_DIRECTORY/postgresql.conf`

Stores configuration parameters to enable the Greenplum Command Center features for Greenplum Database server. These parameters are normally set using the `gpconfig` Greenplum Database management utility.

Command Center Console Parameters

The Command Center Console configuration file is on the Command Center host at

`$GPCC_HOME/webserver/conf/app.conf`.

After editing this file, reload the configuration by restarting the Command Center Console.

```
$ gpcc --stop
$ gpcc --start
```

`appname = gpccws`

The web server binary file. Do not change.

`listentcp4 = [true | false]`

When `true`, the address type is `tcp4`. The default is `true`.

`runmode = [prod | dev | test]`

The application mode, which can be `dev`, `prod` or `test`. The default, `prod`, is the recommended setting. In `dev` and `test` modes Command Center prints more verbose log messages. These are different logs than the logs affected by the `log_level` parameter.

`session = [true | false]`

Use sessions to manage user experience. The default is `true`. Sessions are stored in memory.

`enablexsrf = [true | false]`

Enable CSRF protection.

`xsrffix = <seconds>`

CSRF expire time. The default is `2592000` seconds.

`xsrffkey = <token_string>`

The CSRF token.

```
rendertype = json
```

The render type of web server. Do not change.

```
printallsqls = [true | false]
```

Print all backend gpperfmon SQL to the web server console. The default is `false`.

```
master_port = <port>
```

The Greenplum Database master port. The default is `5432`.

```
log_level
```

The level of messages to log: `Debug`, `Info`, or `Error`. The default is `Info`. The values are not case-sensitive.

```
r
```

```
path = /usr/local
```

Path to the directory where Greenplum Command Center is installed.

```
display_name = <display_name>
```

The display name for the console.

```
enable_kerberos = [true | false]
```

True if Kerberos authentication is enabled for Command Center. The default is `false`.

```
HTTPSCertFile = </path/to/cert.pem>
```

```
HTTPSKeyFile = </path/to/cert.pem>
```

Set both of these properties to the full path to a `.pem` file containing the certificate and private key for the Command Center web server.

```
EnableHTTPS = [true | false]
```

Enable listening on the secure SSL port. True if SSL is enabled. Only one of `EnableHTTPS` or `EnableHTTP` can be true.

```
EnableHTTP = [true | false]
```

Enable listening on the HTTP port. True if SSL is not enabled. Only one of `EnableHTTP` or `EnableHTTPS` can be true.

```
httpsport = <port>
```

The web server port when `EnableHTTPS` is true. The default is `28080`.

```
httpport = <port>
```

The web server port when `EnableHTTP` is true. The default is `28080`.

```
rpcport = <port>
```

The port on which the Command Center backend receives data from metrics collector agents. The default is `8899`.

```
master_host = <hostname>
```

The Greenplum Database host name. The default is `localhost`.

gpmetrics Configuration File Reference

Greenplum Command Center uses the `gpcc.conf` configuration file to save configuration information entered in the Command Center user interface. You should not normally edit the `gpcc.conf` file

directly. Instead, modify configuration information in the Command Center user interface.

The `gpcc.conf` file is created in the `$MASTER_DATA_DIRECTORY/gpmetrics/` directory on the Greenplum Database master or standby host where you start Command Center. The file is an INI-format configuration file, containing properties defined as `key = value` entries, one property per line.

Property	Description
<code>allow_anonymous</code>	If <code>true</code> , Command Center users can access the Query Monitor view without logging into Command Center. You can change this setting on the Command Center Admin> Permissions page.
<code>resource_queue_import_status</code>	Command Center uses this property to determine whether to offer to import Greenplum Database resource queues to resource groups when you access the Admin> Workload Mgmt view. The default is <code>false</code> .
<code>emailFrom</code>	The email address to set on the “From:” line of alert emails. The default is <code>noreply-gpcc-alerts@pivotal.io</code> . Note: Set the email and smtp properties on the Command Center Admin> Alerts page.
<code>emailTo</code>	A comma-separated list of email addresses to send alert emails.
<code>smtpUsername</code>	The account name to use when authenticating with the SMTP server.
<code>smtpServer</code>	The address and port of the SMTP server to use for alert emails.
<code>smtpPassword</code>	The password used to authenticate the SMTP user with the SMTP server, base 64-encoded.

Setup Configuration File

A setup configuration file contains properties used to configure Greenplum Command Center when you perform a non-interactive Command Center installation. The file is passed to the `gpccinstall` command with the `-c` option:

```
$ ./gpccinstall-<version> -c <config_file>
```

The configuration file contains `name: value` or `name=value` entries, one per line. Comments begin with a `#` or `;` character and continue through the end of the line.

See [Installing Pivotal Greenplum Command Center](#) for more information about installing Command Center with a configuration file.

Parameters

`path`

The path to the directory where Greenplum Command Center software will be installed. The directory must be writable by the `gadmin` user on all hosts in the Greenplum Cluster.

`display_name`

The name to display in the Command Center user interface. The default display name is `gpcc`.

`master_port`

The Greenplum Database master port. Default: `5432`.

web_port

The listen port for the Command Center web server. The default is 28080.

enable_ssl

True if client connections to the Command Center web server are to be secured with SSL. The default is **false**. If **true** the `ssl_cert_file` parameter must be set and the SSL certificate must be installed on the host where you run Command Center.

ssl_cert_file

If `enable_ssl` is **true**, set this parameter to the full path to a valid certificate in PEM file format. The certificate must be installed on the host where you run Command Center.

enable_kerberos

Set to **true** to enable Kerberos authentication.

krb_mode

The Kerberos authentication scheme to use. The default is 1.

- **1 - normal mode (default)** - The Command Center Kerberos keytab file contains the Command Center principal and may contain principals for Command Center users. If the principal in the client's connection request is in the Command Center's keytab file, Command Center uses the client's principal for database connections. Otherwise, Command Center uses the gpmon user for database connections.
- **2 - strict mode** - Command Center has a Kerberos keytab file containing the Command Center service principal and a principal for every Command Center user. If the principal in the client's connection request is in the keytab file, the web server grants the client access and the web server connects to Greenplum Database using the client's principal name. If the principal is not in the keytab file, the connection request fails.
- **3 - gpmon_only mode** - Command Center uses the gpmon database role for all Greenplum Database connections. No client principals are needed in the Command Center's keytab file.

webserver_url

The web server hostname, from the Kerberos HTTP service principal.

keytab

Path to the keytab file containing Kerberos principals for the Command Center web server and users.

Examples

```
#####
# GPCC 4.0 setup configuration file
#####
path = /opt
display_name = Greenplum Database Production Cluster
master_port = 5432
webserver_port = 28081
EnableHTTP = true          ; allow both HTTP and HTTPS
EnableHTTPS = true
ssl_cert_file = /etc/ssl/cert.pem
```

```
enable_kerberos = false
```


Securing Greenplum Command Center

Greenplum Command Center Console can be secured by encrypting network traffic between the web server and users' browsers, authenticating Command Center users, and managing users' permissions to access Command Center features.

SSL/TLS Encryption

Greenplum Command Center supports SSL/TLS encryption to secure connections between browsers and the Command Center web server. Command Center supports TLS 1.2 protocol and higher. When enabled, Command Center uses the Secure WebSockets API, enabling long-lived, full-duplex connections, in addition to encryption.

To enable SSL/TLS encryption, you need a combined certificate/key file for the Command Center web server in place when you install Command Center. The file contains a private key and a server certificate.

You can request a key pair and certificate from your organization's internal certificate authority or from an external certificate authority. You can also create a self-signed certificate with a cryptography suite such as OpenSSL. If you create a self-signed certificate, however, clients will have to override a security warning when they first connect to the Command Center web server.

To create the combined certificate/key file, create a text file, for example `server.pem`, and copy the entire body of private key and certificate into it. Make sure to include the beginning and end tags:

```
-----BEGIN RSA PRIVATE KEY-----  
  < private key >  
-----END RSA PRIVATE KEY-----  
-----BEGIN CERTIFICATE-----  
  < certificate >  
-----END CERTIFICATE-----
```

You can concatenate additional certificates to the end of the file if a certificate chain is required to authenticate the server certificate.

Place the file on the server where Command Center will execute, for example in the `/etc/ssl/certs` directory of the Greenplum Database master host. When you install Command Center the installer prompts you to enter the full path to this file. See [Command Center Console Parameters](#) for details.

Authentication Options

Users logging in to Greenplum Command Center are authenticated with the Greenplum Database host-based authentication system. Users can enter credentials as a user name and password or, if Kerberos authentication is configured, by authenticating with Kerberos on their workstation before browsing to the Command Center web server.

Note: Greenplum Command Center does not accept logins from the gpadmin user, or from users configured with trust authentication in the `pg_hba.conf` file.

Database users must first be added to the Greenplum Database by using commands such as `CREATE ROLE` or `CREATE USER`. The `LOGIN` privilege is required. This example creates a login user with an encrypted password:

```
CREATE ROLE cc_user WITH LOGIN ENCRYPTED PASSWORD 'changeme';
```

The `pg_hba.conf` configuration file determines how authentication will proceed. This file contains a list of entries that are compared to attributes of the user's connection request, including the type of connection, network location of the originating host, database name, and login user name. When a match is found, the authentication method specified in the entry is applied.

The `pg_hba.conf` file can be viewed by Operators and edited by Admins in the Command Center console on the [Admin>Authentication](#) page.

The `md5` and `password` authentication methods authenticate the user name and password with the Greenplum Database `pg_roles` system table. The `md5` method requires the password to be MD5-encoded when sent over the network, so it is preferred over the `password` method, which sends the password in clear text.

The `ldap` authentication method authenticates the user name and password with an LDAP server. The LDAP server and parameters are specified in the options field of the `pg_hba.conf` entry. See the PostgreSQL [LDAP authentication](#) documentation for the format of the LDAP options.

The `gss` authentication method is used for Kerberos authentication. To use Kerberos with Command Center, Kerberos authentication must be enabled for the Greenplum Database system and Command Center must also be configured. Users authenticate with the Kerberos KDC on their workstations (using `kinit`, for example) before connecting to the Command Center web server. The role name in Command Center is the user's Kerberos principal name.

For details about setting up Kerberos authentication, see [Enabling Kerberos Authentication with Greenplum Command Center](#).

See the PostgreSQL [Authentication methods](#) documentation for additional details of the authentication options.

Authorization

Note: The functionality described in this section has not been fully implemented in Greenplum Command Center 4.0.0. Only Admin and Self Only permission levels are available.

Command Center manages permission levels using Greenplum Database roles and groups. The Basic, Operator Basic, and Operator permission levels correspond to the `gpcc_basic`, `gpcc_operator_basic`, and `gpcc_operator` group roles in the database. The Admin permission level is conferred to roles that have the `SUPERUSER` privilege. A user who has not been added to any of the groups and does not have `SUPERUSER` privilege has the most restrictive permission level, Self Only.

Greenplum Database superusers can manage permission levels on the Command Center [Admin>Permissions](#) page. Superusers can also directly assign users roles in the database by using the `ALTER USER`, `ALTER GROUP`, and related commands to add or remove users from groups and add

or remove the `SUPERUSER` privilege. If a role is configured for more than one permission level, Command Center uses the highest permission level.

Command Center users have the following capabilities, according to their permission levels:

Self Only

Users can view metrics and view and cancel their own queries.

Any Greenplum Database user successfully authenticated through the Greenplum Database authentication system can access Greenplum Command Center with Self Only permission. Higher permission levels are required to view and cancel other's queries and to access the System and Admin Control Center screens.

Basic

Allows users to view metrics, view all queries, and cancel their own queries.

Users with Basic permission are members of the Greenplum Database `gpcc_basic` group.

Operator Basic

Allows users to view metrics, view their own and others' queries, cancel their own queries, and view the System and Admin screens.

Users with Operator Read-only permission are members of the Greenplum Database `gpcc_operator_basic` group.

Operator

Allows users to view their own and others' queries, cancel their own and other's queries, and view the System and Admin screens.

Users with Operator permission are members of the Greenplum Database `gpcc_operator` group.

Admin

Allows users access to all views and capabilities in the Command Center.

Greenplum Database users with the `SUPERUSER` privilege in Greenplum Database have Superuser permissions in Command Center.

Managing Greenplum Command Center Authentication

The **Admin > Authentication** screen allows users with Operator Basic, Operator, and Admin permission to view the Greenplum Database host-based authentication file, `pg_hba.conf`.

Host-Based Authentication Controls user access for all GPDB activity on this server

For guidance on managing `pg_hba.conf` consult the postgres documentation

pg_hba.conf This version 2018-09-25, 12:10 LOAD VERSION... ABANDON CHANGES SAVE CONFIG AND UPDATE GPDB

Type	Database	User	Address (CIDR/Hostname)	Method	Comment
local	all	gpadmin		ident	
host	all	gpadmin	127.0.0.1/28	trust	
host	all	gpadmin	192.168.1.144/32	trust	
host	all	gpadmin	::1/128	trust	
host	all	gpadmin	2605:a601:4199:bb00:a00:27ff:...	trust	
host	all	gpadmin	fe80::a00:27ff:fe6c:43b4/128	trust	
local	replication	gpadmin		ident	
host	replication	gpadmin	sameonet	trust	
local	gpperfmon	gpmon		md5	
host	all	gpmon	127.0.0.1/28	md5	
host	sales	nickd	nickd-pc	md5	
host	all	gpmon	::1/128	md5	

Type Database User Address (CIDR/Hostname) Method Comment

host all gpmon 192.168.1.144/28 md5

DONE Cancel

Users with Admin permission can add, remove, change, and move entries in the file. The Command Center UI validates entries to ensure correct syntax. Previous versions of the file are archived so that you can restore an earlier version or audit changes.

See [Authentication Options](#) for an overview of user authentication options for Greenplum Database and Greenplum Command Server.

See [Configuring Client Authentication](#) in the *Greenplum Database Administrator Guide* for a detailed description of the contents of the `pg_hba.conf` file.


Viewing the Host-Based Authentication File





Choose **Admin>Authentication** to display the content of the Greenplum Database `pg_hba.conf` file.

The `pg_hba.conf` file contains a list of entries that specify the characteristics of database connection requests and authentication methods. When Greenplum Database receives a connection request from a client, it compares the request to each entry in the `pg_hba.conf` entry in turn until a match is found. The request is authenticated using the specified authentication method and, if successful, the connection is accepted.

Editing the Host-Based Authentication File

Command Center users with the *Admin* permission can edit the `pg_hba.conf` file. Note that any changes you make are lost if you move to another screen before you save them.

- To change an existing entry, click anywhere on the entry. Edit the fields and click **Save** to save your changes, or **Cancel** to revert changes.
- To move an entry up or down in the list, click on the  symbol, drag the line to the desired location, and release.
- To add a new entry to the end of the file, click **Add New Entry** at the bottom of the screen. Edit the fields and click **Save** to save your changes, or **Cancel** to abandon the new entry.

- To add a new entry after an existing entry, highlight the existing entry and click . Edit the fields and click **Save** to save your changes, or **Cancel** to abandon the new entry.
- To copy an entry, select the entry and click . A copy of the selected entry is added below the selected entry and displayed for editing. Edit the fields and click **Save** to save your changes, or **Cancel** to abandon the copy.
- To add a comment to the file, add an entry by clicking **Add New Entry** or  and then choose # from the **Type** list.
- To toggle an entry between active and inactive, select the line and click the **active/inactive** toggle control to the right. This action adds or removes a comment character (#) at the beginning of the entry.
- To remove an entry, highlight the line and click . The entry is displayed with strikethrough text. You can restore the entry by highlighting it and clicking **undelete**. The entry is permanently removed when you click **Save config and update GPDB**.
- To finish editing, click **Save config and update GPDB**. Then click **Save and Update** to save your changes or click **Cancel** to return with your edits intact.

When you select **Save and Update**, the `pg_hba.conf` file is saved and refreshed in Greenplum Database. Note that existing client connections are unaffected.

Loading a Previous Version of the Host-Based Authentication File

When you save a new version of the `pg_hba.conf` file, a copy is saved in the Greenplum Database `$MASTER_DATA_DIRECTORY/pg_hba_archive` directory as `pg_hba.conf-<timestamp>`.

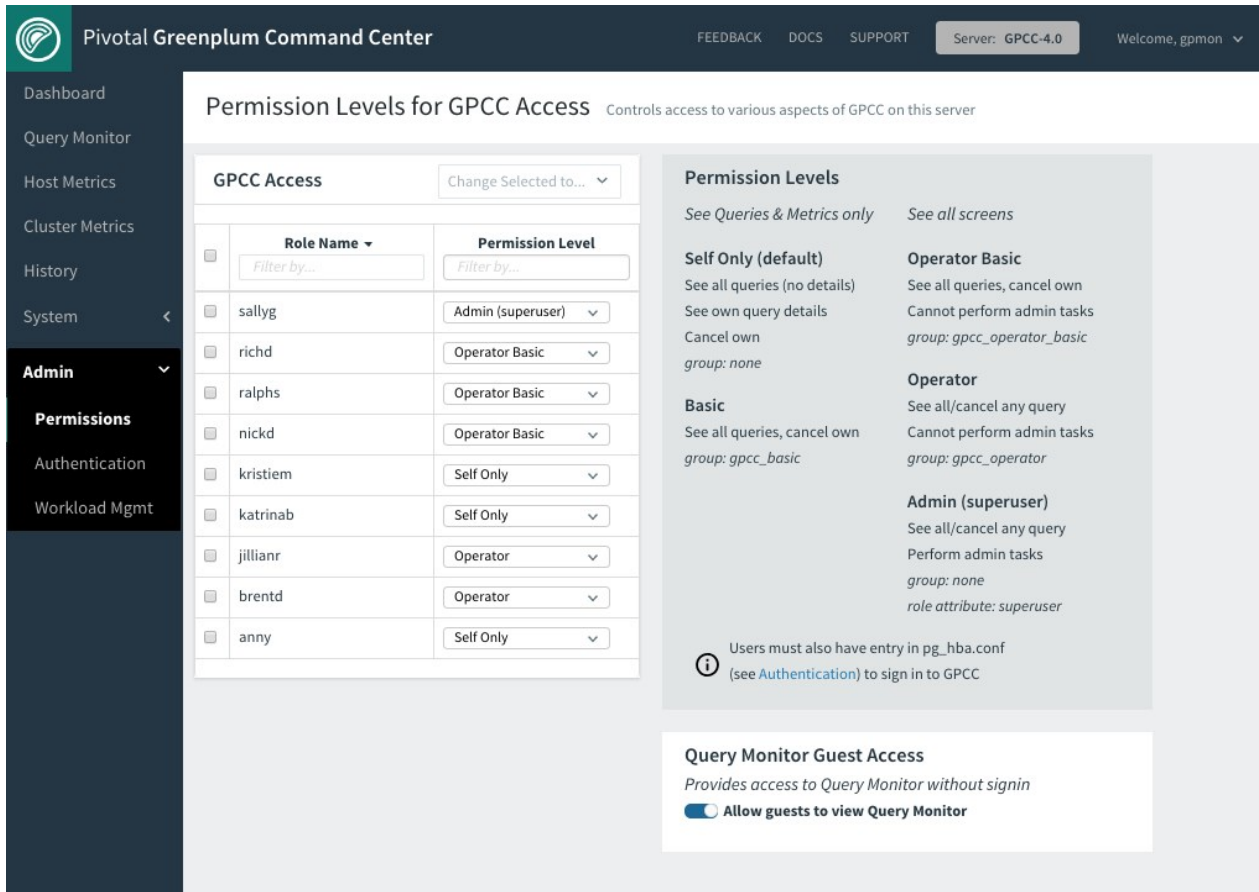
To view an archived version of the `pg_hba.conf` file, click **Load versions...** and click the timestamp for the version to display.

To revert to a previous version of the file, load the previous version and then click **Save config and update GPDB**. The configuration is refreshed in Greenplum Database and saved as a new version in the archive directory.

Managing Greenplum Command Center Permissions

The **Permissions Levels for GPCC Access** screen allows users with Operator Basic, Operator, or Admin permission to view Command Center user permissions for Greenplum Database users. Users with Admin permission can set permissions for any user.

Users with Operator Basic, Operator, and Admin permission can also see if the Guest Access to Query Monitor feature is enabled or disabled, and Admin users can toggle this feature on and off.



Viewing User Permissions

Initially, all Greenplum Database login users are included in the list of roles with their current permission levels.

- To filter by role name, enter all or part of the user’s database role name in the *Role Name* field. The filter performs a simple substring search and displays users with matching role names. Click the **Role Name** label to reverse the search order.
- To filter for users with a specific permission level, choose the permission level from the **Permission Level** list.
- Role Name and Permission Level filters can be used together.
- To reset the filters, remove all text from the *Role Name* field and choose **Filter by...** from the **Permission Level** list.

Changing User Permission Levels

Users with Admin permission can change permission levels.

1. Use the **Role Name** and **Permission Level** filters to display the roles you want to change.
2. Check the box next to a role name to select the user, or check the box in the heading to select all displayed users.
3. Select the new permissions level for each user from the list in the **Permission Level** column, or select a new permission level for all selected users from the **Change Selected to...** list.

Enabling or Disabling Guest Access to Query Monitor

When enabled, the **Guest Access to Query Monitor** feature allows anyone with access to the Greenplum Command Center web server to click **View Query Monitor** on the Command Center sign-in screen and see the **Query Monitor** page without logging in. These anonymous users cannot cancel queries and cannot access any other Command Center features.

When this feature is off, the **View Query Monitor** link does not appear on the sign-in screen and anonymous users cannot see the **Query Monitor** page.

Command Center users with Admin permission can toggle the feature on an off with a mouse click. Users with Operator or Operator Basic permission see a message reporting if the feature is on or off.

Securing the gpmon Database User

The Greenplum Database gpmon user is a superuser role used to manage the gpperfmon database. The `gpperfmon_install` utility, which must be run before you install Greenplum Command Center Console, creates the gpmon role.

Greenplum Database uses the gpmon role to update the gpperfmon database with data collected by agents running on the segment hosts. The Command Center web server uses the gpmon role to connect to the gpperfmon database as well as databases monitored by the Command Center.

When `gpperfmon_install` creates the gpmon role, it prompts for a password, which it then adds to the `.pgpass` file in the gpadmin user's home directory. The entry in the `.pgpass` file is similar to the following:

```
*:5432:gpperfmon:gpmon:changeme
```

See [The Password File](#) in the PostgreSQL documentation for details about the `.pgpass` file.

The `.pgpass` file is required on the Greenplum Database master host to start the gpperfmon data collection agents. If you run Greenplum Command Center on a different host, you can copy the `.pgpass` file to that host, or you can run the Command Center `gpcc` management utility with the `-W` option to request password entry each time you start or stop Command Center or request status.

In the `$MASTER_DATA_DIRECTORY/pg_hba.conf` authentication file, `gpperfmon_install` creates these entries:

```
local    gpperfmon          gpmon          md5
host     all              gpmon          127.0.0.1/28   md5
host     all              gpmon          ::1/128        md5
```

If you authenticate users with Kerberos, you can also set up Kerberos authentication for the gpmon role on the Greenplum master and standby hosts. Kerberos authentication is supported with TCP connections only; `local` entries use Linux sockets and authenticate with the `.pgpass` file password, even if you have enabled Kerberos for `host` entries.

Changing the gpmon Password

To change the gpmon password, follow these steps:

1. Log in to Greenplum Database as a superuser and change the gpmon password with the `ALTER ROLE` command:

```
# ALTER ROLE gpmon WITH ENCRYPTED PASSWORD 'new_password';
```

2. On the Greenplum master host, update the password in the `.pgpass` file in the gpadmin home directory (`~/pgpass`). Replace the existing password in the line or lines for gpmon with the new password.

```
*:5432:gpperfmon:gpmon:new_password
```

3. Ensure that the `.pgpass` file is owned by gpadmin and RW-accessible by gpadmin only.

```
$ chown gpadmin:gpadmin ~/.pgpass
$ chmod 600 ~/.pgpass
```

4. Restart Greenplum Command Center with the `gpcc` utility.

```
$ gpcc stop
$ gpcc start
```

Be sure to also update the `.pgpass` file on the standby master host.

Authenticating gpmon with Kerberos

If you authenticate Greenplum Database and Command Center users with Kerberos, you can also authenticate the gpmon user with Kerberos.

To prepare for installing Command Center with Kerberos authentication, follow these steps:

1. Create the gpperfmon database using the Greenplum Database `gpperfmon-install` management utility. See [Creating the gpperfmon Database](#).
2. On the KDC, create a keytab file containing the Kerberos principal for the gpmon user, just as you would for any Kerberos-authenticated client. Install the file on the Greenplum master and standby hosts.
3. Update the entries for gpmon in the `$MASTER_DATA_DIRECTORY/pg_hba.conf` file to use the `gss` authentication method.

```
host all gpmon 0.0.0.0/0 gss include_realm=0 krb_realm=GPDB.EXAMPLE.COM
```

Note that `local` entries in `pg_hba.conf` cannot be authenticated with Kerberos. If there is a `local` entry for the gpmon user, it will use the `.pgpass` file to authenticate with the database. See [The pg_hba.conf file](#) in the PostgreSQL documentation for complete `pg_hba.conf` file documentation.

4. Log in to the master host as gpadmin and authenticate the gpmon user.

```
$ kinit gpmon
```

5. Install Greenplum Command Center to set up the Kerberos-enabled Command Center.

Enabling Authentication with Kerberos

If you have enabled Kerberos authentication for Greenplum Database, you can set up Greenplum Command Center to accept connections from Kerberos-authenticated users.

Greenplum Database and Command Center include support for the Generic Security Service Applications Program Interface (GSS-API) standard. A related standard, Simple and Protected GSS-API Negotiation Mechanism (SPNEGO), describes the protocol GSS-API clients and servers use to agree on the method of authentication.

With a SPNEGO-compliant web application such as Command Center, the client and server agree on the authentication method on the client's initial HTTP request. If Kerberos authentication is not supported on both ends of the connection the server falls back to basic authentication, and displays a login form requesting a user name and password. If a user has authenticated on the workstation with Kerberos and has a valid ticket granting ticket, the web browser offers the user's credential to the Command Center web server. A Kerberos-enabled Command Center web server is configured to handle the authenticated user's connection request in one of three modes, called strict, normal, or gpmon-only.

Strict

Command Center has a Kerberos keytab file containing the Command Center service principal and a principal for every Command Center user. If the principal in the client's connection request is in the keytab file, the web server grants the client access and the web server connects to Greenplum Database using the client's principal name. If the principal is not in the keytab file, the connection request fails.

Normal

The Command Center Kerberos keytab file contains the Command Center principal and may contain principals for Command Center users. If the principal in the client's connection request is in Command Center's keytab file, it uses the client's principal for database connections. Otherwise, Command Center uses the gpmon user for database connections.

gpmon-only

Command Center uses the gpmon database role for all Greenplum Database connections. No client principals are required in the Command Center's keytab file. This option can be used, for example, if Command Center users authenticate with Active Directory and you do not want to maintain client principals in the keytab file.

If you have set up Kerberos authentication for Greenplum Database, most of the configuration required to enable Command Center Kerberos authentication has been done. The Command Center Kerberos configuration builds upon the Greenplum Database Kerberos setup.

Kerberos authentication can be enabled by responding to prompts when you install Command Center, or you can use the `gpcc --krbenable` command to enable Kerberos after Command Center has been installed.

Before You Begin

Kerberos authentication must be enabled for Greenplum Database. See [Using Kerberos Authentication](#) for instructions. Make sure the following prerequisites are met before you continue:

- The `krb5-workstation` package and associated libraries (`libkrb5*`) must be installed on the

Greenplum master host and each client workstation.

- The date and time on the Greenplum master host and all client workstations must be synchronized with the KDC.
- The `krb5.conf` configuration file must be the same on the KDC host, the Greenplum Database master host, and client workstations.
- The KDC database must have a service principal for Greenplum Database. The default service name for Greenplum Database is `postgres/<master-host>@<realm>`. You can choose a service name other than `postgres`, but it must match the value of the `krb_srvname` parameter in the `$MASTER_DATA_DIRECTORY/postgresql.conf` file.
- A keytab file with the Greenplum Database principal must be installed on the Greenplum master host and identified by the `krb_server_keyfile` parameter in the `$MASTER_DATA_DIRECTORY/postgresql.conf` file.
- Each client workstation can have a keytab file containing their Kerberos principal, `<username>@<realm>`.

Add Command Center Principals to the KDC Database

Before you configure Command Center for Kerberos authentication, you must create the required Kerberos principals. All of the principals used with Command Center are created in the Greenplum Database Kerberos realm. Command Center users can use the same Kerberos principal to authenticate with Command Center and Greenplum Database.

Command Center Service Principal

A service principal is needed for the Command Center web server. This principal has the format `HTTP/<host>@<realm>`. For example, if users access Command Center at the URL `http://mdw.example.com:28080`, the `<host>` part of the service key is `mdw.example.com` and the `<realm>` part is the Greenplum Database Kerberos realm, for example `GPDB.KRB`.

Note that Kerberos authentication only works if Command Center users enter the host in the same format specified in the Kerberos service principal. If the principal specifies the FQDN, for example, using the host's IP address in the browser URL will not work; the web server will fall back to basic authentication.

Greenplum Database gpmon User

Command Center uses the gpmon Greenplum role to access the gpperfmon database, which contains data presented in the Command Center UI.

You can choose to authenticate the gpmon user with Kerberos or with basic authentication. To use Kerberos, you must create a principal for the gpmon user.

If you choose to use basic authentication you do not need a Kerberos principal for the gpmon user. The gpmon user will authenticate with Greenplum Database using the password saved in the `.pgpass` file in the gpadmin user's home directory on the host running Command Center. If you run Command Center on a host other than the Greenplum Database master host, you must copy the `.pgpass` file from the master host to the Command Center host. See [Changing the gpmon Password](#) for instructions to manage the gpmon password.

Command Center Users

Add Kerberos principals for any Command Center users who do not already have principals in the KDC for Greenplum Database.

Adding Kerberos Principals

To add the required principals, perform the following steps as root on the KDC server.

1. Add a principal for the Command Center web service. Be sure to specify the `<gpcc-host>` in the same format that users should enter the host in their browsers.

```
# kadmin.local -q "addprinc -randkey HTTP/<gpcc-host>@<realm>"
```

2. If you want the gpmon database user to use Kerberos authentication, add a gpmon principal.

```
# kadmin.local -q "addprinc gpmon@<realm>"
```

3. Add principals for any new Command Center users.

```
# kadmin.local -q "addprinc cc_user1@<realm>"
```

Repeat for each new Command Center user.

Set Up Keytab Files

After you have created all of the Kerberos principals needed, you create and distribute keytab files. Keytab files contain Kerberos principals and encrypted keys based on the principals' Kerberos passwords. Keytab files are needed for the Greenplum Database master and standby hosts and the Command Center host.

You can also create a keytab file for each Greenplum Database or Command Center user containing just the user's principal. This keytab file is installed on the user's workstation to enable the user to authenticate to Kerberos. Note that all keytab files must contain the most recent versions of the principals' keys.

Command Center Running on the Greenplum Master Host

If the Greenplum Command Center web server is running on the Greenplum Database master host, Command Center can share the Greenplum Database keytab file. You need to create a keytab file that contains the following principals:

- Service key for the `postgres` process on the Greenplum Database master host, for example `postgres/mdw.example.com@GPDB.EXAMPLE.`
- Service key created for Command Center in the previous section, for example `HTTP/mdw.example.com@GPDB.KRB.`
- A principal for every Kerberos-authenticated Greenplum Database or Command Center user.

All service keys and principals should be in the Greenplum Database realm.

To create a keytab file for Greenplum Database and Command Center, perform the following steps as root on the KDC server.

1. Create a keytab file containing the Greenplum Database service key, the command center service key, and all database and Command Center users.

```
kadmin.local -q "ktadd -k gpdb-kerberos.keytab postgres/mdw.example.com@GPDB.KRB
B HTTP/mdw.example.com@GPDB.KRB"
```

You can enter one or more principals with each `ktadd` command. You can specify a wildcard using the `-glob` option. For example this command adds all principals in the `GPDB.KRB` realm, including service principals and admin users.

```
kadmin.local -q "ktadd -k gpdb-kerberos.keytab -glob *@GPDB.KRB"
```

2. Copy the keytab you created to the Greenplum Database master host, replacing the old keytab file. The location of the file is given by the `krb_server_keyfile` parameter in the `$MASTER_DATA_FILE/postgresql.conf` file. Set the permissions on the file so that it can be read only by the `gpadmin` user.
3. Update any entries required for new Greenplum Database principals in the `pg_hba.conf` file and `pg_ident.conf` files. See [Update the Greenplum Database pg_hba.conf File](#) for details.

Command Center Running on the Standby Master

If the Command Center web server is on a different host than the Greenplum Database master, you need separate keytab files for Greenplum Database and Command Center. The keytab file for Greenplum Database may not require any updates, but you will need to create a keytab file for Command Center.

- The Greenplum Database keytab file must contain the Greenplum Database service key and all principals for users with database access.
- The Command Center keytab file contains the Command Center service key and principals for users that have Command Center access. Users with Command Center access must also have Greenplum Database access, so user principals in the Command Center keytab file must also be in the Greenplum Database keytab file.

Update the Greenplum Database keytab if you created new database roles and principals for Command Center. For example, if you want to use Kerberos authentication for the `gpmon` user, you must create a principal and add it to both the Greenplum Database and Command Center keytab files.

To create the keytab file for Command Center, perform the following steps as root on the KDC host.

```
...
```

1. Create a keytab file and add the Command Center service key.

```
kadmin.local -q "ktadd -k gpcc-kerberos.keytab HTTP/smdw.example.com@GPDB.KRB"
```

2. If you want to authenticate the `gpmon` user with Kerberos, add the `gpmon` principal.

```
kadmin.local -q "ktadd -k gpcc-kerberos.keytab gpmon@GPDB.KRB"
```

3. Add principals for all Command Center users:

```
kadmin.local -q "ktadd -k gpcc-kerberos.keytab cc_user1@GPDB.KRB cc_user2@GPDB.KRB"
```

You can enter one or more principals with each `ktadd` command.

4. Enter `quit` to exit `kadmin.local`.
5. Copy the keytab you created to the the host running Command Center, for example:

```
$ scp gpcc-kerberos.keytab gpadmin@<host-name>:/home/gpadmin
```

6. Update any entries required for new principals in the `pg_hba.conf` file and `pg_ident.conf` files on the Greenplum master. See [Update the Greenplum Database pg_hba.conf File](#).

Update the Greenplum Database pg_hba.conf File

The Greenplum Database `$MASTER_DATA_DIRECTORY/pg_hba.conf` configuration file determines which authentication methods to use to allow database access.

If you created new Command Center users, you may need to add an entry to allow access via Command Center. The entry for an individual user has this format:

```
host database <user-name> <gpcc CIDR> gss [options]
```

Authentication for the `gpmon` user needs to be set up in the `pg_hba.conf` file in one of the following ways.

Basic authentication

The `/home/gpadmin/.pgpass` file contains the password for `gpmon` to use. See [Changing the gpmon Password](#) for details. An entry in the `pg_hba.conf` file specifies the md5 authentication method for `gpmon`:

```
local all gpmon md5
```

Trust authentication

On the Greenplum Database master host only, the `gpmon` user can access databases without authentication:

```
local all gpmon trust
```

The `/home/gpadmin/.pgpass` file is not needed.

Kerberos authentication

A Kerberos principal has been created for the `gpmon` user and added to the Greenplum Database and Command Center keytab files.

```
host all gpmon <gpcc CIDR> gss [options]
```

Remove any existing reject rules for `gpmon`:

```
host all gpmon <auth-method> reject
```

See [Using Kerberos Authentication](#) for more information about the `pg_hba.conf` file.

Enable Kerberos for Command Center

Set up Command Center to use the Command Center keytab file you created.

If you are adding Kerberos authentication to an existing Command Center, use the `gpcc` command. For example:

```
$ gpcc --krbenable
```

Enter the Command Center host name and path to the keytab file at the prompts. See the [gpcc Reference](#) for more information.

Authenticating With Kerberos on the Client Workstation

To use Kerberos Command Center authentication, the user must have authenticated with Kerberos using the `kinit` command-line tool.

The user then accesses the Command Center web server with a URL containing the host name in the format specified in the Command Center service principal and the port number, for example `http://mdw.example.com:28080`.

The user's web browser must be configured to use the SPNEGO protocol so that it offers the user's Kerberos principal to the web browser. The method for configuring web browsers varies with different browsers and operating systems. Search online to find instructions to enable SPNEGO with your browser and OS.

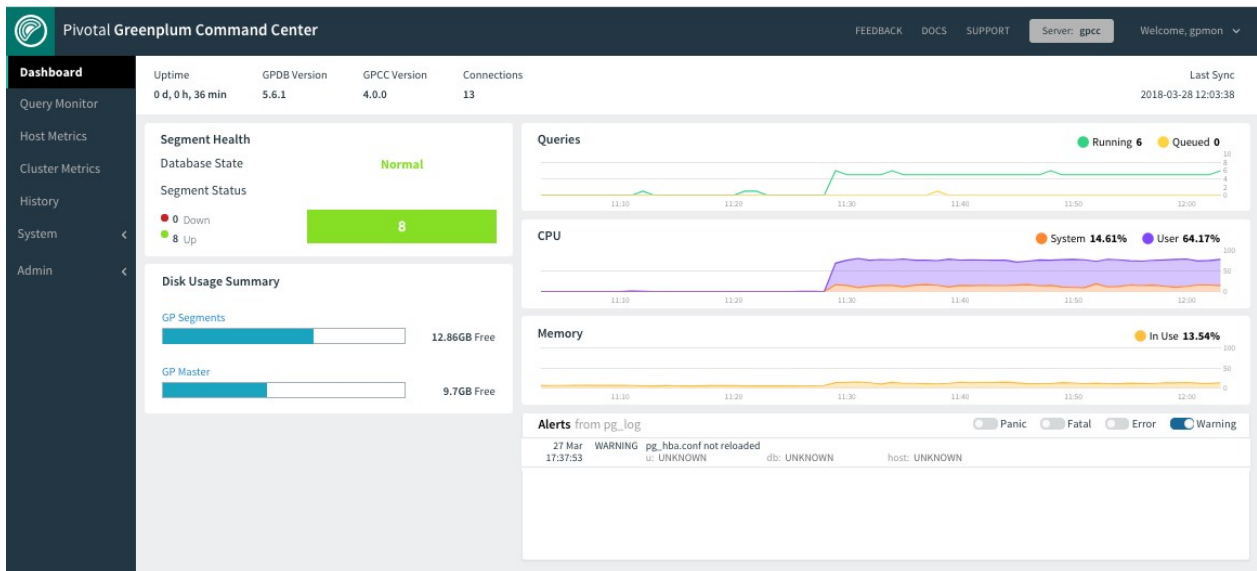
Monitoring the Greenplum Database System

- [Dashboard](#)
- [Cluster State](#)
- [Segment Status](#)
- [Cluster Metrics](#)
- [Host Metrics](#)
- [Storage Status](#)

Dashboard

The **Dashboard** displays when you first sign in to Pivotal Greenplum Command Center. The **Dashboard** provides a quick view of the current system status, Segment Health, Queries, CPU, Memory, and Disk usage.

Clicking on a panel provides more detailed information about the metric. The Alerts panel shows the most recent messages from the Greenplum Database log file. Some information is available only to Command Center users with Admin or Operator permission level.



System Information

The following system information is displayed at the top of the page.

Uptime

The elapsed time since the Greenplum Database system was last started.

GPDB Version

The version of the Greenplum Database software the monitored cluster is running.

GPCC Version

The version of the Greenplum Command Center software.

Connections

The number of active Greenplum Database sessions (client connections).

Server

The display name for this Greenplum Command Center.

Last Sync

Date and time the data was last synchronized. The Command Center user interface updates views with live data every 15 seconds.

Segment Health

The **Segment Health** section of the Dashboard provides a quick overview of the status of the database system and segments this Command Center monitors.

Database State

Database State is the current state of the Greenplum Database system. Following are some possible database states:

- **Normal:** The database is functioning with no major errors or performance issues.
- **Segment(s) Down:** The database is in change-tracking mode or resync mode. Overall performance and system reliability is greatly reduced. See the *Pivotal Greenplum Database System Administrator Guide* for information about resolving this condition.
- **Standby Down:** The master standby instance cannot be reached.
- **Database Unreachable:** The Greenplum Performance Monitor agent cannot connect to the database. The database is likely down. See the *Pivotal Greenplum Database System Administrator Guide* for troubleshooting information.
- **Unbalanced:** Some segments are not running in their preferred roles. That is, primaries are running as mirrors and mirrors are running as primaries, resulting in unbalanced processing.
- **Resyncing:** The database is performing a recovery or rebalance operation.

An error message or state other than the above may be an indication of a network connectivity problem, or of another undetected problem in the Greenplum Database system. Run the ``gpstate`` utility on the master host to determine if there are issues to troubleshoot in the Greenplum Database system.

Segment Status

The bar graph in the **Segment Status** section shows the up or down status of all database segments in your Pivotal Greenplum Database system. A color indicator and associated number indicate the number of database segments that are currently in that particular state. Segments can have the following states:

- **Up** (Green)

- **Down** (Red)

Clicking the **Segment Status** panel displays the [Segment Status](#) Command Center page.

Disk Usage Summary

This chart displays total disk usage and disk available for the Greenplum master host and segment hosts at the last synchronization. Hover over the chart to see the amount of disk used, free, and total.

Queries

This graph displays a summary view of active and queued queries for the last 60 minutes. Click on the colored dot next to the **Running** or **Queued** label to toggle the line on or off. At least one line must be visible at all times. Hover over the graph to display the number of queries for each visible line at that point in time.

CPU

This graph displays average CPU usage across the entire cluster, for the last 60 minutes. The graph displays separate lines for system processes and user processes. The user CPU usage includes the Greenplum database master, standby, and segment processes. Click on the colored dot next to the **System** or **User** label to toggle that line on or off. At least one line must be visible at all times.

Hovering the cursor over a line in the graph displays a small window with the percentage of CPU used at that point in time for the visible lines and the total if both the system and user lines are visible.

Memory

This graph displays the average percent of memory used across the entire cluster over the last 60 minutes. Hover over the line to display the percent of memory used at that point in time.

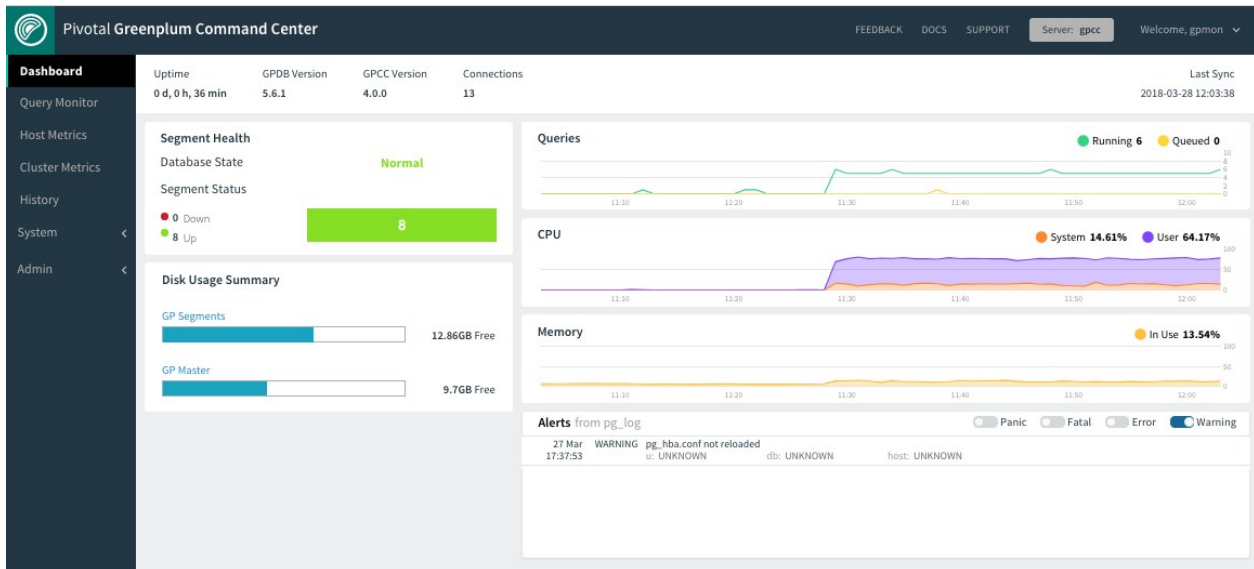
Alerts

Admin and Operator permission levels only

The **Alerts** panel displays recent messages from the Greenplum Database `pg_log` log file. The panel is updated at each synchronization. Filter the messages by severity level using the controls at the top right of the panel.

Greenplum Database Cluster State

The Greenplum Command Center **Dashboard** provides a quick view of the current system status, Segment Health, Queries, CPU, Memory, and Disk usage. Clicking on a panel provides more detailed information about that metric. The **Alerts** panel shows the most recent messages from the Greenplum Database log file. Some information is available only to Command Center users with Admin or Operator permission level.



System Information

The following system information is displayed at the top of the page.

Uptime

The elapsed time since the Greenplum Database system was last started.

GPDB Version

The version of the Greenplum Database software the monitored cluster is running.

GPCC Version

The version of the Greenplum Command Center software.

Connections

The number of active Greenplum Database sessions (client connections).

Server

The display name for this Greenplum Command Center.

Last Sync

Date and time the data was last synchronized. The Command Center user interface updates views with live data every 15 seconds.

Segment Health

The **Segment Health** section of the Dashboard provides a quick overview of the status of the database system and segments this Command Center monitors.

Database State

Database State is the current state of the Greenplum Database system. The state can be one of the following:

- **Normal:** The database is functioning with no major errors or performance issues.
- **Segment(s) Down:** The database is in change-tracking mode or resync mode. Overall performance and system reliability is greatly reduced. See the *Pivotal Greenplum Database System Administrator Guide* for information about resolving this condition.

- **Database Unreachable:** The Greenplum Performance Monitor agent cannot connect to the database. The database is likely down. See the *Pivotal Greenplum Database System Administrator Guide* for troubleshooting information.
- **Unbalanced:** Some segments are not running in their preferred roles. That is, primaries are running as mirrors and mirrors are running as primaries, resulting in unbalanced processing.
- **Resyncing:** The database is performing a recovery or rebalance operation.

Segment Status

The bar graph in the **Segment Status** section shows the up or down status of all database segments in your Pivotal Greenplum Database system. A color indicator and associated number indicate the number of database segments that are currently in that particular state. Segments can have the following states:

- **Up** (Green)
- **Down** (Red)

Clicking the **Segment Status** panel displays the [Segment Status](#) Command Center page.

Disk Usage Summary

This chart displays total disk usage and disk available for the Greenplum master host and segment hosts at the last synchronization. Hover over the chart to see the amount of disk used, free, and total.

Queries

This graph displays a summary view of active and queued queries for the last 60 minutes. Click on the colored dot next to the **Running** or **Queued** label to toggle the line on or off. At least one line must be visible at all times. Hover over the graph to display the number of queries for each visible line at that point in time.

CPU

This graph displays average CPU usage across the entire cluster, for the last 60 minutes. The graph displays separate lines for system processes and user processes. The user CPU usage includes the Greenplum database master, standby, and segment processes. Click on the colored dot next to the **System** or **User** label to toggle that line on or off. At least one line must be visible at all times.

Hovering the cursor over a line in the graph displays a small window with the percentage of CPU used at that point in time for the visible lines and the total if both the system and user lines are visible.

Memory

This graph displays the average percent of memory used across the entire cluster over the last 60 minutes. Hover over the line to display the percent of memory used at that point in time.

Alerts

Admin and Operator permission levels only

The **Alerts** panel displays recent messages from the Greenplum Database `pg_log` log file. The panel is updated at each synchronization. Filter the messages by severity level using the controls at the top right of the panel.

Segment Status

The **Segment Status** page provides a health overview for the Greenplum Database segments and details for each primary and mirror segment.

Segment Status Role and health related information Last Sync
2018-03-28 12:49:28

Segment Summary		Segment Health		
Database State	Total Segments	Status	Replication Mode	Preferred Role
Normal	8	0 Down 8 Up	0 Not Syncing 0 Resyncing 0 Change Tracking 8 Synced	0 Not Preferred 8 Preferred
Mirrors Acting as Primary	Segment Hosts			
0	2			
Recommended Actions				
None				

Hostname	Address	Port	DBID	Content ID	Status	Role	Preferred Role	Replication Mode	Last Event [Total]
sdw1	sdw1	40000	2	0	UP	Primary	Primary	Synced	
sdw2	sdw2	40000	4	2	UP	Primary	Primary	Synced	
sdw1	sdw1	40001	3	1	UP	Primary	Primary	Synced	
sdw2	sdw2	40001	5	3	UP	Primary	Primary	Synced	
sdw2	sdw2	50000	6	0	UP	Mirror	Mirror	Synced	
sdw2	sdw2	50001	7	1	UP	Mirror	Mirror	Synced	
sdw1	sdw1	50000	8	2	UP	Mirror	Mirror	Synced	
sdw1	sdw1	50001	9	3	UP	Mirror	Mirror	Synced	

Segment Summary

Greenplum Database is most efficient when all segments are operating in their preferred roles. The **Segment Summary** panel tells you the overall segment status and if any mirrors are acting as primaries.

The **Segment Summary** panel provides the following information:

Database State

The database state can be one of the following:

- **Normal:** The database is functioning with no major errors or performance issues.
- **Segment(s) Down:** The database is in change-tracking mode or resync mode. Overall performance and system reliability is greatly reduced. See the *Pivotal Greenplum Database System Administrator Guide* for information about resolving this condition.
- **Database Unreachable:** The Greenplum Performance Monitor agent cannot connect to the database. The database is likely down. See the *Pivotal Greenplum Database System Administrator Guide* for troubleshooting information.
- **Unbalanced:** Some segments are not running in their preferred roles. That is, primaries are running as mirrors and mirrors are running as primaries, resulting in

unbalanced processing.

- **Resyncing:** The database is performing a recovery or rebalance operation.

Mirrors Acting as Primary

The number of mirror segments acting as primary segments.

Recommended Actions

Suggests actions to perform to restore the cluster to balance. These include:

- Recover and Rebalance
- Rebalance

These actions are executed from the command line using the `gprecoverseg` Greenplum management utility. See `gprecoverseg` in the *Pivotal Greenplum Database Utility Reference* for more information.

Total Segments

The total number of primary and mirror segments in the Greenplum cluster.

Segment Hosts

The total number of segment hosts in the Greenplum cluster.

Segment Health

The **Segment Health** panel contains charts for Greenplum Database segments' status, replication mode, and preferred roles.

Status

Numbers of segments that are down and up.

Replication Mode

A chart that shows the number of segments in each of the possible replication modes.

- **Not Syncing:** The primary segment and mirror segment are active and all changes to the primary segment have been copied to the mirror using a file block replication process.
- **Change Tracking:** If a primary segment is unable to copy changes to its mirror segment using the file replication process, it logs the unsent changes locally so they can be replicated when the mirror again becomes available. This can happen if a mirror segment goes down or if a primary segment goes down and its mirror segment automatically assumes the primary role.
- **Resyncing:** When a down segment is brought back up, administrators initiate a recovery process to return it to operation. The recovery process synchronizes the segment with the active primary and copies the changes missed while the segment was down.
- **Synced:** Once all mirrors and their primaries are synchronized, the system state becomes synchronized.

Preferred Roles

The red portion of the Preferred Role chart shows the numbers of segments that not operating in

their preferred primary or mirror roles. If the chart is not solid green, the performance of the Greenplum cluster is not optimal.

Primary and mirror segments are distributed evenly among the segment hosts to ensure that each host performs an equivalent share of the work and primary segments and their mirror segments reside on different segment hosts. When a primary segment goes down, its mirror on another host in the cluster automatically assumes the primary role, increasing the number of primary segments running on that host. This uneven distribution of the workload will affect query performance until the down segment is restored and the segments are returned to their original, preferred, roles.

Segment Table

The table at the bottom of the **Segment Status** page contains a detailed row for every primary and mirror segment in the Greenplum Cluster. The table has the following columns for each segment:

Hostname

The name of the segment host where the segment is running.

Address

The network interface on the segment host for the segment.

Port

The port number assigned to the segment.

DBID

The unique identifier for the segment instance.

ContentID

The content identifier for the segment, from 0 to the number of segments minus 1. A primary segment and its mirror have the same ContentID. The master and standby master, which have ContentID -1, are excluded from the table.

Status

“UP” if the segment is running, “DOWN” if the segment has failed or is unreachable.

Role

The segment’s current role, either “primary” or “mirror”.

Preferred Role

The segment’s intended role, either “primary” or “mirror”.

Replication Mode

The replication status for the segment. See [Segment Health](#) for possible values.

Last Event|[Total]

The date and time of last segment health-related activity. Click to display a list of recent events.

Cluster Metrics

The **Cluster Metrics** page shows consolidated statistics for all segment hosts in the Greenplum cluster. Master and standby master hosts are excluded from the metrics.



The charts display metrics for the last time period set by the control in the top right corner of the screen.

Use the **Show/hide Charts** control to choose which metrics to display.

Hover over any of the charts to see values for the metrics at a point in time in pop-up boxes. The charts are synchronized so that hovering over any chart shows the same point in time in all charts.

The current value of a metric is shown in the upper right corner of its chart.

On charts with multiple metrics, toggle the display for a line on or off by clicking the line's label in the legend at the top right of the chart. At least one line must be displayed. All lines are redisplayed at the next quantum interval.

The page has charts for the following metrics:

Queries

The number of queries running and the number of queries queued to run.

CPU

The percentage CPU used by system processes and the percentage CPU used by user processes.

Memory

Percentage of memory in use.

Memory is calculated as follows:

Total = MemTotal

Free = MemFree + Buffers + Cached

Used = MemTotal - Free

Disk I/O

Disk read and write rates in megabytes per second.

Network

Network I/O read and write rates in megabytes per second. Network metrics include traffic over all NICs (network interface cards), including internal interconnect and administrative traffic.

Load

System load average for 1-minute, 5-minute, and 15-minute periods.

Swap

Percentage of swap space used.

Host Metrics

The **Host Metrics** page displays a table of the hosts in the cluster with statistics collected at the most recent quantum interval.

Hostname	CPU Total/Sys/User (%)	Memory In Use (%)	Disk R (MB/s) Skew	Disk W (MB/s) Skew	Net R (MB/s) Skew	Net W (MB/s) Skew
sdw1	83.15	14.60	0	1.57	1.32	1.96
sdw2	78.25	13.47	0	3.72	1.33	1.97
mdw	51.70	9.69	0	0.01	2.94	1.67

At the top, **Last Sync** displays the time the statistics were last updated.

Click a column header to sort the table by that column. Click again to toggle between ascending and descending sort. Master and standby hosts are not included in the sort and are always displayed following the sorted list of segment hosts.

For each server, the following columns are displayed:

Hostname

The hostname name of the server.

CPU Total/Sys/User (%)

The total percentage of CPU in use is displayed next to a graph illustrating the CPU used for system and user processes. Hover over the table cell to show the percentages used for system and user processes and the percentage CPU idle.

Memory In Use (%)

The percentage of host memory in use is displayed next to a graph illustrating the memory in use and available. Hover over the table cell to see memory used and available in gigabytes.

Memory is calculated as follows:

Total = MemTotal

Free = MemFree + Buffers + Cached

Used = Total - Free

Disk R (MB/s) | Skew

Disk read rate in megabytes per second is displayed next to a graph of calculated disk read skew. Hover over the table cell to see a Low/Medium/High rating for disk skew.

Disk W (MB/s) | Skew

Disk write rate in megabytes per second is displayed next to a graph of calculated disk write skew. Hover over the table cell to see a Low/Medium/High rating for disk write skew.

Net R (MB/s) | Skew

Network read rate in megabytes per second is displayed next to a graph of calculated network

read skew. Hover over the table cell to see a Low/Medium/High rating for network read skew.
 Net W (MB/s) | Skew

Network write rate in megabytes per second is displayed next to a graph of calculated network write skew. Hover over the table cell to see a Low/Medium/High rating for network write skew.

About Skew Calculations

Disk and Network skew ratings are calculated as each server’s standard deviation from the mean calculated from all segment hosts.

Low

Value is within 1 standard deviation from the mean. (Note: if the variance of the set is less than 3, skew is considered low regardless of deviation from mean.)

Moderate

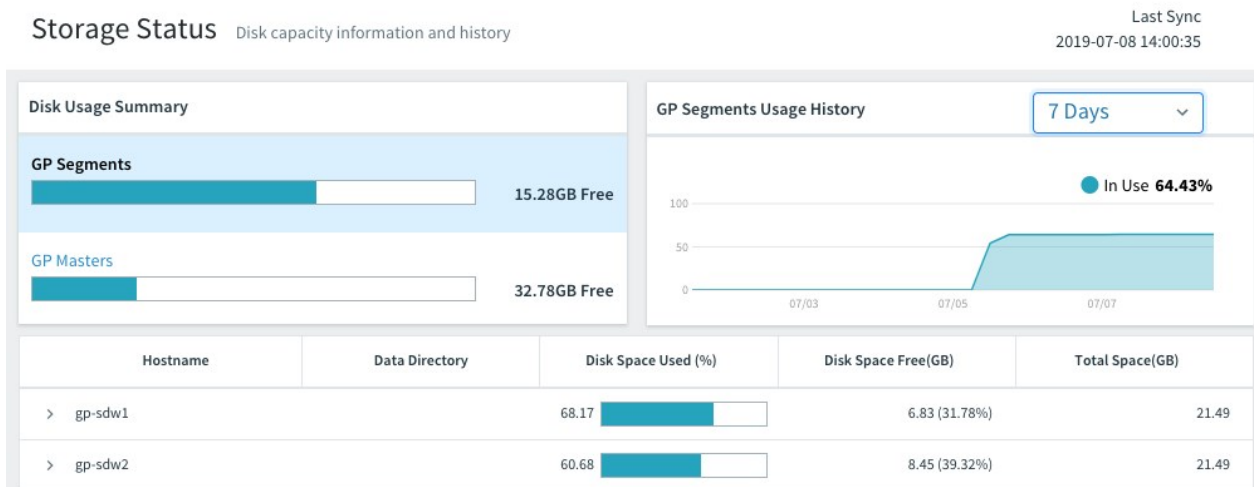
Value is between 1 and 2 standard deviations from the mean.

Very High

Value is greater than 3 standard deviations from the mean.

Storage Status

The **Storage Status** page shows current historical disk usage for Greenplum master and segment hosts.



Disk Usage Summary

You can see current disk space in use, space free, and total space in the **Disk Usage Summary** panel. Disk space metrics for the segment hosts (**GP Segments**) and the master (**GP Masters**) are shown in separate bar charts.

The **GP Segments** bar chart shows combined disk space for all segments.

The **GP Masters** bar chart shows combined disk space for master and standby master.

Click either of the summary charts to see details by host: space used, free, and total in gigabytes and as a percentage of the total.

Click a host name to display a list of directories on the host file system where partitions containing

one or more Greenplum Database data directories are mounted. Hover over the directory name to see a list of the Greenplum data directories the partition contains.

GP Segments Usage History

The **GP Segments Usage History** panel presents a chart of percentage of disk space in use for the time period set by the control in the panel header.

Hover over the chart to see the percentage disk in use by all Greenplum Database segments at any given point in time.

GP Masters Usage History

The **GP Masters Usage History** panel presents a chart of percentage of disk space in use by the master and standby masters for the time period set by the control in the panel header.

Hover over the chart to see the percentage disk in use at any given point in time.

Monitoring and Managing Greenplum Database Queries

- Query Monitor
- Query Details
- Query History

Query Monitor

The **Query Monitor** view allows you to view information for all Greenplum Database server queries, including details about queries running, queued to run, and blocked by other queries. Users with Admin or Operator permission can see and cancel all users' queries.

The screenshot displays the Pivotal Greenplum Command Center interface. The main content area is titled "Query Monitor" and shows "Current queries by all users". At the top right, it indicates the "Current Time" as 2018-07-11 15:42:05. Below the title, there is a summary bar showing "25 Running", "0 Queued", and "2 Blocked" queries. To the right of this bar are buttons for "CANCEL QUERY" and "EXPORT".

Query ID	Status	User	Database	Workload	Submitted	Queued Time	Run Time	Spill Files	Blocked by
1531266727-13954-2	Running	gpadmin	gpadmin	admin_gr...	15:35:48	4m 53s	1m 35s	--	--
1531266727-13956-2	Running	tpch_1	gpadmin	default_g...	15:35:54	4m 50s	1m 32s	352.80 MB	--
1531266727-13968-2	Running	tpch_1	gpadmin	default_g...	15:36:20	4m 24s	1m 32s	179.79 MB	--
1531266727-14005-2	Running	tpch_1	gpadmin	default_g...	15:39:01	1m 43s	1m 32s	158.94 MB	--
1531266727-14011-2	Running	tpch_4	gpadmin	benchmark	15:39:34	1m 11s	1m 31s	216.40 MB	--
1531266727-14033-2	Running	tpch_1	gpadmin	benchmark	15:40:12	32s	1m 32s	102.81 MB	--
1531266727-14034-2	Running	tpch_1	gpadmin	benchmark	15:40:13	1s	2m 2s	102.72 MB	--
1531266727-14035-2	Running	tpch_1	gpadmin	benchmark	15:40:13	4s	1m 59s	406.96 MB	--
1531266727-14036-2	Running	tpch_1	gpadmin	benchmark	15:40:13	2s	2m 1s	143.46 MB	--
1531266727-14037-2	Running	tpch_1	gpadmin	benchmark	15:40:13	2s	2m 1s	98.08 MB	--
1531266727-14038-2	Running	tpch_1	gpadmin	benchmark	15:40:13	2s	2m 1s	129.43 MB	--
1531266727-14039-2	Running	tpch_4	gpadmin	benchmark	15:40:14	31s	1m 31s	95.54 MB	--
1531266727-14040-2	Running	tpch_4	gpadmin	benchmark	15:40:14	2s	2m 0s	135.91 MB	--
1531266727-14041-2	Running	tpch_1	gpadmin	benchmark	15:40:14	1s	2m 1s	99.39 MB	--

If a Command Center administrator has enabled Query Monitor Guest Access, anyone able to access the Command Center web server can view the system status and query list on this page without signing in to Command Center. Anonymous users, however, cannot cancel queries or access any other Command Center features.

With the information available in this view, Greenplum Database administrators can easily:

- Understand how the system is being used — both in real-time and trending over time.

- Identify and diagnose problem queries while they are running, detect skew, find runaway queries, and so on.
- Review and balance the query load on the system by better optimizing and scheduling the query load.
- Cancel queries that disrupt system performance.

Note

The Query Monitor does not display queries executed by the `gpmon` user in the `gpperfmon` database.

Query Metrics

The Query Monitor table displays the following columns for queries.

Query ID

An identification string for the query. If the column is blank, no query ID has been assigned yet. In the Console, this looks like “1295397846-56415-2”. Command Center generates this ID by combining the query record’s `tmid`, `ssid`, and `ccnt` fields.

- `tmid` is a time identifier for the query.
- `ssid` is the session id.
- `ccnt` is the number of the command within the session.

Status

The status of the query. This can be one of the following:

- Queued: the query has not yet started to execute
- Running: execution has started, but is not yet complete
- Blocked: the query is waiting for one or more other queries to release locks
- Done: completed successfully
- Cancelling: cancel request sent, cancel pending
- Cancelled: terminated, no longer running
- Idle Transaction: the transaction is open, but idle, for example, waiting while a user in an interactive session enters a statement

User

The Greenplum Database role that submitted the query.

Database

The name of the database that was queried.

Workload

The resource group or resource queue that is managing the query.

Submitted

The time the query was submitted to the query planner.

Queued Time

The amount of time the query has been (or was) in queue awaiting execution.

Run Time

The amount of time since query execution began.

Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

Blocked by

Shows the number of locks blocking the query. Hover over the column to display details of the locks. The tip shows the type of each lock, the ID of the transaction or query that holds the lock, the Greenplum Database role holding the lock, and the amount of time the query has been blocked.

4s	--	--	1 Blockers
0s	1m 14s		Access Exclusive 1531266727-14058-2 User gpadmin
0s	1m 7s	60	Query blocked for 1m 27s
1s	1m 6s	52	

Using the Query Monitor Controls

- Click a column heading to sort the rows on that column in ascending or descending order.
- Click the checkbox at the left of a row to choose a query to cancel or export. Click the checkbox in the heading row to choose all queries.
- Click **Cancel Query** to cancel selected queries. A pop-up box prompts you to enter a reason. You can enter a message of up to 128 characters to display with the error message that is received by users whose queries are cancelled.
- Click **Export** to download a comma-separated values (CSV) text file containing rows for the selected queries. When no queries are selected, all rows are exported. The default file name is `spreadsheet.csv`.
- Click any query ID to see the [Query Details](#), including metrics, the text of the query, and the query plan.

Query Details

The **Query Details** view displays query metrics, the text of the query, and the query plan and progress for a single query selected from the [Query Monitor](#) view.

Query ID: 1527709348-39-3 ● Running Run Time 4m 9s CANCEL QUERY Current Time 2018-05-30 14:47:53 help

Details	Current Performance	Blocking (0)	Blocked by (0)
User: tpch_4	CPU: 11.08 %	CPU Skew: 41.67 %	
Database: gpadmin	Memory: 511.64 MB	Spill Files: 608.69 MB	
Res Queue: pg_default	Disk R: 0.00 MB/s	Disk W: 2.06 MB/s	
Submitted: 14:43:51			
Queued Time: 0s			
Run Time: 4m 9s			

Query Text

```

select
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice,
  sum(l_quantity)
from
  customer,
  orders,
  lineitem
where
  o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 5
  )
  and c_custkey = o_custkey
  and o_orderkey = l_orderkey
group by
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice
order by
  o_totalprice desc,
  o_orderdate
;
                
```

Plan & Progress Textual Plan

```

Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..6554.33 rows=2402130 width=49)
Merge Key: orders.o_totalprice, orders.o_orderdate
-> GroupAggregate (cost=0.00..6160.61 rows=600533 width=49)
  Group By: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
  -> Sort (cost=0.00..6115.27 rows=600533 width=48)
    Sort Key: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
    -> Hash Join (cost=0.00..2977.88 rows=600533 width=48)
      Hash Cond: public.lineitem.l_orderkey = orders.o_orderkey
      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
      -> Hash (cost=1888.16..1888.16 rows=124319 width=41)
        -> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..1888.16 rows=124319 width=41)
          Hash Key: orders.o_orderkey
          -> Hash Join (cost=0.00..1872.21 rows=124319 width=41)
            Hash Cond: orders.o_custkey = customer.c_custkey
            -> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..1371.15 rows=124319 width=22)
              Hash Key: orders.o_custkey
              -> Hash EXISTS Join (cost=0.00..1362.59 rows=124319 width=22)
                Hash Cond: orders.o_orderkey = public.lineitem.l_orderkey
                -> Table Scan on orders (cost=0.00..459.64 rows=374619 width=22)
                -> Hash (cost=784.20..784.20 rows=124319 width=4)
                  -> Result (cost=0.00..784.20 rows=124319 width=4)
                    Filter: (sum(public.lineitem.l_quantity)) > 5::numeric
                    -> HashAggregate (cost=0.00..773.98 rows=310798 width=12)
                      Group By: public.lineitem.l_orderkey
                      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
                    -> Hash (cost=434.95..434.95 rows=37596 width=23)
                      -> Table Scan on customer (cost=0.00..434.95 rows=37596 width=23)
                Settings: optimizer=on
                Optimizer status: PQO version 2.58.0
                
```

Query Execution Metrics

The Query ID, execution status, and run time of the query are displayed at the top.

The following metrics are displayed for the query.

User

The Greenplum Database role that submitted the query.

Database

The name of the database that was queried.

Workload

The name of the resource group or resource queue that is managing the query.

Planner

The name of the query planner used for this query, GPORCA or Legacy.

Submitted

The time the query was submitted to the query planner.

Queued Time

The amount of time the query has been (or was) in queue awaiting execution.

Run Time

The amount of time since query execution began.

Est. Progress

An estimate of the percentage of the query execution completed. The estimate is calculated from row count and cost estimates generated by either the GPORCA or legacy planner for the particular query and the available statistics. The estimate does not account for the many other factors that can affect query execution so it should not be seen as a reliable predictor of query completion time.

The progress for each plan node is calculated as the ratio of actual rows produced to the planner's estimate of the total number of rows the node will produce:

$$NodeProgress = \frac{ActualRows}{EstimatedRows}$$

The overall progress for the query is estimated using the calculated node progress and the planner's cost estimates:

$$OverallProgress = \frac{\sum NodeProgress(x) * Est.Cost(x)}{\sum Est.Cost(x)}$$

If the estimate is greater than 100% and the query has not yet completed, 99.9% completion is reported. 100% is reported if the formula produces an estimated percentage greater than 100%.

CPU Master

Current CPU percent on the Greenplum Database master host for this query.

CPU Segments

(Active queries only.) Current CPU percent average for all segment processes executing this query. The percentages for all processes running on each segment are averaged, and then the average of all those values is calculated to render this metric. Current CPU percent average is always zero in historical and tail data. The master and standby master are excluded from the calculation.

CPU Time

Total CPU time consumed by all processes on all segments executing this query.

CPU Skew

The amount of CPU skew. CPU skew occurs when query executor processes for one segment use a disproportionate amount of CPU compared to processes for other segments executing the query. This value is calculated as

$$1 - (average_segment_CPU / maximum_segment_CPU)$$

Memory

Memory consumed by all segment processes executing the query.

Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

Disk R

The current average disk read rate for all segment hosts.

Disk W

The current average disk write rate for all segment hosts.

Locks and Blocks

Contains two lists of locks currently blocking transactions. Click a list to expand and view the contents.

- A list of locks held by this query, including the type of each lock and the queries blocked by that lock.

Locks and Blocks

Holding **1** lock and block **3** query

Access Exclusive	1531266727-15345-2
--	1531266727-15341-2
	1531266727-15342-2

Blocked by **12** query

- A list of queries that hold locks that block this query and the lock type.

Locks and Blocks

Holding **1** lock and block **3** query

Blocked by **12** query

1531266727-15295-2	Access Share
1531266727-15157-2	Access Share
1531266727-15160-2	Access Share

Query Text and Execution Plan

The query text and the query’s plan and execution progress are shown in the lower panels of the Query Details view. The text of the query is displayed in the left panel, and the plan and progress is displayed in the right panel.

Query Text

The **Query Text** panel displays the text of the query as it was submitted to Greenplum Database.

Command Center can display up to 100K characters. If you click **COPY**, up to 100K characters of the query text are copied to the clipboard.

If the query text is longer than 100K characters, a message is displayed with a link you can use to download the full text of the query. The name of the text file is the ID of the query with a `.txt` extension. The file is available to download for 24 hours after the query completes, or until the query has been saved to history, once history collection is enabled.

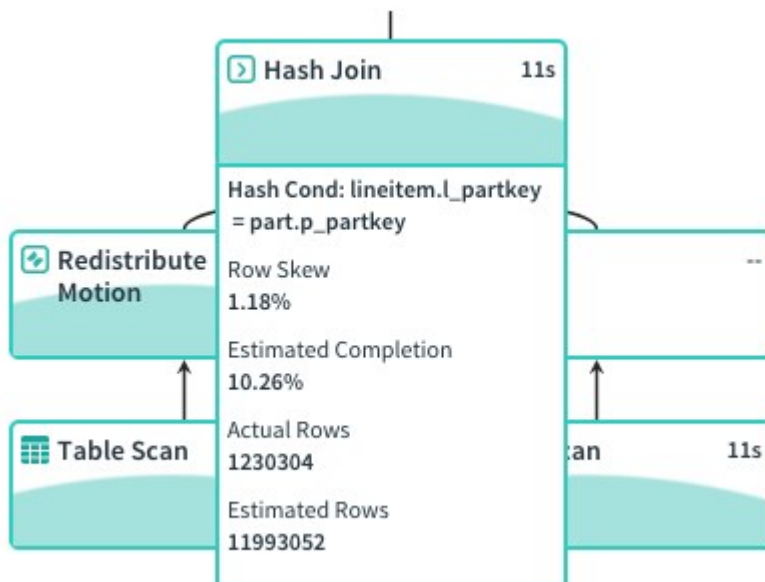
Query Plan and Progress

The **Plan & Progress** tab in the lower right panel is a graphical representation of the query plan with animation and real-time metrics to show execution progress. Each box in the tree represents a step in the query execution plan. The boxes are labeled with the operation they represent and have a CPU usage metric. Query execution begins at the bottom of the tree and ends at the top.

Before a step begins to execute, the box has a solid white fill. When the step becomes active, the box is animated with a green and white fill to indicate that the operator is working. When the step has completed, the box has a solid green fill.

Query execution plans are executed in “slices,” portions of the query plan that segments can work on independently in parallel. The plan is sliced wherever a data motion operator occurs. The time displayed in the upper right corner of each box is the amount of CPU time used for all steps in the slice to which the step belongs. The visual query plan does not illustrate slices, but you can find information about slices in the textual plan.

If you click a step, the box expands to show additional details.



The expanded plan box contains the following metrics.

- The type of operator. When the operator is a table scan, the name of the scanned table is included. See [Query Plan Execution](#) for descriptions of the operators.
- Information related to the current operation, such as the hash key, merge key, join condition, or filter condition.
- Row Skew - the amount of row skew for the current operator, a value from 0% to 100%. Row skew occurs when some segments process more rows than other segments. The percentage is calculated as $(1 - (\text{average_segment_rows} / \text{maximum_segment_rows})) * 100$.
- Estimated Completion - the current percentage of actual rows to estimated rows for this plan step. The percentage can exceed 100% if the operation produces more rows than the optimizer’s estimate. The percentage changes to “Completed” when the operation completes.
- Actual Rows - The current number of rows produced by this step. Note that for nested join operators, the Actual Rows is estimated since the actual row counts are not available while

the join is executing.

- Estimated Rows - The estimated number of rows the operator will produce.

Textual Plan

Select the **Textual Plan** tab and click **RUN EXPLAIN** to generate the text representation of the explain plan.

The **RUN EXPLAIN** button is dimmed if Command Center is unable to generate the explain plan. Command Center is unable to generate the explain plan if the size of the query text is greater than 100K characters or if the query text contains multiple statements.

Query ID: 1527709348-39-3 ● Running Run Time 4m 9s

CANCEL QUERY
Current Time
2018-05-30 14:47:53
help

Details	Current Performance		Blocking (0)	Blocked by (0)
User: tpch_4	CPU: 11.08 %	CPU Skew: 41.67 %		
Database: gpadmin	Memory: 511.64 MB	Spill Files: 608.69 MB		
Res Queue: pg_default	Disk R: 0.00 MB/s	Disk W: 2.06 MB/s		
Submitted: 14:43:51				
Queued Time: 0s				
Run Time: 4m 9s				

Query Text

```

select
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice,
  sum(l_quantity)
from
  customer,
  orders,
  lineitem
where
  o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 5
  )
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice
order by
  o_totalprice desc,
  o_orderdate
;
                    
```

Plan & Progress BETA
Textual Plan

```

Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..6554.33 rows=2402130 width=49)
Merge Key: orders.o_totalprice, orders.o_orderdate
-> GroupAggregate (cost=0.00..6160.61 rows=600533 width=49)
  Group By: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
  -> Sort (cost=0.00..6115.27 rows=600533 width=48)
    Sort Key: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
  -> Hash Join (cost=0.00..2977.88 rows=600533 width=48)
    Hash Cond: public.lineitem.l_orderkey = orders.o_orderkey
    -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
    -> Hash (cost=1888.16..1888.16 rows=124319 width=41)
      -> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..1888.16 rows=124319 width=41)
        Hash Key: orders.o_orderkey
        -> Hash Join (cost=0.00..1872.21 rows=124319 width=41)
          Hash Cond: orders.o_custkey = customer.c_custkey
          -> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..1371.15 rows=124319 width=22)
            Hash Key: orders.o_custkey
            -> Hash EXISTS Join (cost=0.00..1362.59 rows=124319 width=22)
              Hash Cond: orders.o_orderkey = public.lineitem.l_orderkey
              -> Table Scan on orders (cost=0.00..459.64 rows=374619 width=22)
              -> Hash (cost=784.20..784.20 rows=124319 width=4)
                -> Result (cost=0.00..784.20 rows=124319 width=4)
                  Filter: (sum(public.lineitem.l_quantity)) > 5::numeric
                  -> HashAggregate (cost=0.00..773.98 rows=310798 width=12)
                    Group By: public.lineitem.l_orderkey
                    -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
                -> Hash (cost=434.95..434.95 rows=37596 width=23)
                  -> Table Scan on customer (cost=0.00..434.95 rows=37596 width=23)
                    
```

Settings: optimizer=on
Optimizer status: PQO version 2.58.0

The textual plan is the output of the Greenplum Database **EXPLAIN** command for the query. The query plan steps are labeled with arrows (**->**) and the structure of the query plan tree is indicated with indentation.

The **Optimizer status:** line at the bottom of the textual plan reports whether the explain plan was generated using the GPORCA optimizer (PQO) or the legacy query optimizer.

For help reading the textual explain plan see the **EXPLAIN** command in the *Greenplum Database Reference Guide* and **Query Profiling** in the *Greenplum Database Administrator Guide*. See **Query Execution** for descriptions of the query operators.

VMware, Inc

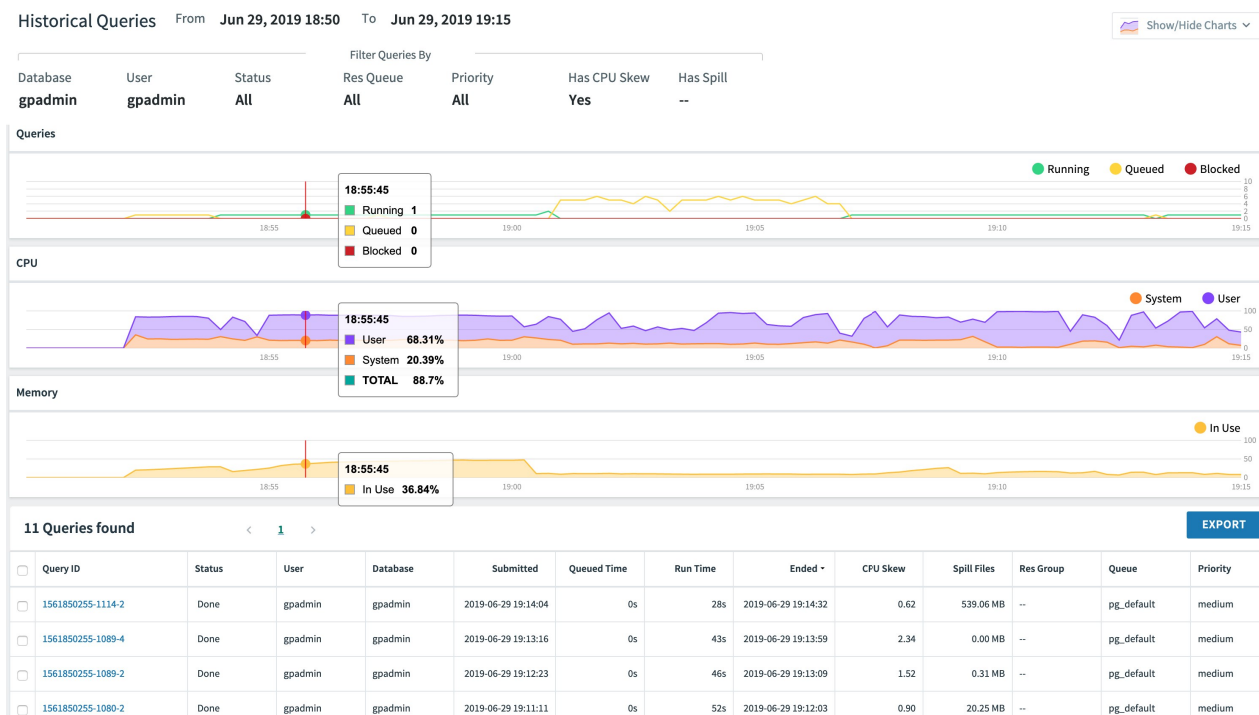
90

History

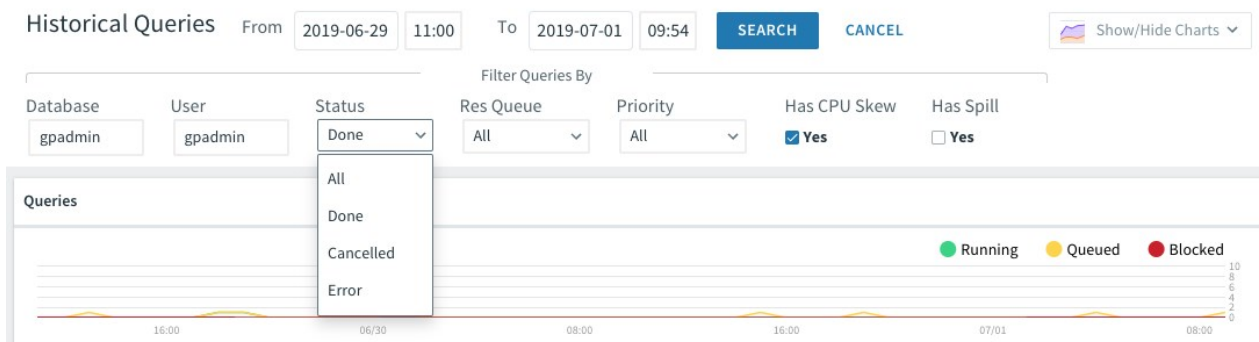
The **History** page allows you to display system metrics and queries executed during a specified time period.

On the [Admin > Settings](#) page you can enable saving the real-time metrics collected by the Greenplum Database metrics collector extension to history in the gpmetrics schema of the gpperfmon database. When you enable collecting this history, the Command Center History, Query Monitor, and Query Detail views all use data derived from the same data collection method.

If you disable GPCC query history collection on the **Admin > Settings** page, the history views display data collected and saved in the gpperfmon database by the `gpmmmon` and `gpsmon` agents. That data is not derived from the real-time metrics displayed on the Query Monitor and Query Detail view. Variations should be expected due to the different data collection methods.



Set the time period to display by entering dates and times in the **From** and **To** date and time fields. You can enter dates by typing them into the date field or by choosing from the pop-up calendar. Enter 24-hour times in HH:MM format.



In the **Filter Queries By** section you can restrict queries that display in the query table at the bottom of the page.

- **Database** - Choose only queries that ran in a specified database.

- **User** - Choose only queried executed by a specified user.
- **Status** - Choose queries that completed with the specified status: *Done*, *Cancelled*, or *Error*.
- **Res Queue** or **Res Group** - Choose queries executed by a specified resource queue or resource group.
- **Priority** - Choose queries that executed with a relative CPU priority: *min*, *low*, *medium*, *high*, or *max*. This option is only available when Greenplum Database is using resource queue-based resource management.
- **Has CPU Skew** - Choose queries that completed with CPU skew greater than zero.
- **Has Spill** - Choose queries that generated spill files.

Click **Search** to display results that match your choices.

Hover over a timeline to see the metrics at that point in time for all unhidden charts.

You can click and drag on a chart to zoom in on a time range. Click **Search** to update the query list and charts to the selected range.

Scroll charts left or right by hovering over the edge of the chart and clicking an arrow. Click < or > to move in half steps. Click « or » to move in full steps.

Charts of the following metrics are available. Show or hide them at any time with the checklist at the upper right of the view.

Queries

The number of queries running, the number of queries queued to run, and the number of queries blocked.

CPU

The percentage of CPU used by system processes and the percentage of CPU used by user processes.

Memory

Percentage of memory in use.

Disk I/O

Disk read and write rates in megabytes per second.

Network

Network I/O read and write rates in megabytes per second. Network metrics include traffic over all NICs (network interface cards), including internal interconnect and administrative traffic.

Load

System load average for 1-minute, 5-minute, and 15-minute periods.

Swap

Percentage of swap space used.

Query Metrics

The Query table displays queries that were active during the specified time period, including queries that started before or finished after the specified time. However, queries that are still active are not included in the table; these queries can be viewed on the [Query Monitor](#) page.

The query table has the following columns:

Query ID

An identification string for the query. In the Console, this looks like “1295397846-56415-2”.

Status

The final status of the query. This can be one of the following:

- Done
- Cancelled
- Error

User

The Greenplum Database user who submitted the query.

Database

The name of the database that was queried.

Submitted

The time the query was submitted to the query planner.

Queued Time

The time the query waited before it was executed. In addition to time in the queue, this includes other time such as time in the optimizer.

Run Time

The amount of time the query required to produce a result.

Ended

The time the query completed or was cancelled.

CPU Skew

The amount of CPU skew. CPU skew occurs when query executor processes for one segment use a disproportionate amount of CPU compared to processes for other segments executing the query. This value is the coefficient of variation for the CPU used by processes running this query on each segment, multiplied by 100. For example, a value of .95 is shown as 95.

Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

Res Group or Res Queue

The name of the resource group or resource queue for the query.

Priority

(Res Queue only) A query executes with the CPU priority assigned to its resource queue.

For more information about Resource Queues, Resource Groups, and Query Plans, refer to the *Greenplum Database Administrator Guide*.

Query History Details

The **Query History Details** view displays query metrics, the text of the query, and the query plan and execution statistics for a single query selected from the [Query History](#) view.

Query ID: 1561850255-1133-2 • Done Current Time 2019-06-29 19:23:30

Details				Performance				Locks and Blocks	
User	gpadmin	Submitted	00:14:59	CPU Master	CPU Segments	CPU Time	CPU Skew	Holding 0 lock and block 0 query	
Database	gpadmin	Queued Time	0s	4.21 %	88.42 %	00:11:02	0.47 %	Blocked by 0 query	
Res Queue	pg_default	Run Time	3m 00s	Memory	Spill Files	Disk R	Disk W		
Planner	GPORCA			1.38 GB	1.05 GB	341.33 MB	4.67 GB		

Query Text

```
with frequent_ss_items as
(select substr(i_item_desc,1,30) itemdesc,i_item_sk,item_sk,d_date sold
date,count(*) cnt
 from store_sales
 ,date_dim
 ,item
 where ss_sold_date_sk = d_date_sk
 and ss_item_sk = i_item_sk
 and d_year in (1999,1999+1,1999+2,1999+3)
 group by substr(i_item_desc,1,30),i_item_sk,d_date
 having count(*) >4),
max_store_sales as
(select max(sales) tpcds_cmax
 from (select c_customer_sk,sum(ss_quantity*ss_sales_price) csales
 from store_sales
 ,customer
 ,date_dim
 where ss_customer_sk = c_customer_sk
 and ss_sold_date_sk = d_date_sk
 and d_year in (1999,1999+1,1999+2,1999+3)
 group by c_customer_sk) x),
```

Plan & Progress Textual Plan

Query History Metrics

The Query ID, execution status, and run time of the query are displayed at the top.

The following metrics are displayed for the query.

User

The Greenplum Database role that submitted the query.

Database

The name of the database that was queried.

Res Group / Res Queue

The name of the resource group or resource queue that is managing the query.

Planner

The name of the query planner used for this query, GPORCA or Legacy.

Submitted

The time the query was submitted to the query planner.

Queued Time

The time the query waited before it was executed. In addition to time in the queue, this includes other time such as time in the optimizer.

Run Time

The amount of time the query executed.

CPU Master

The CPU percent on the Greenplum Database master host for this query.

CPU Segments

CPU percent average for all segment processes executing this query. The percentages for all processes running on each segment are averaged, and then the average of all those values is calculated to render this metric. The master and standby master are excluded from the

calculation.

CPU Time

Total CPU time consumed by all processes on all segments executing this query.

CPU Skew

The amount of CPU skew. CPU skew occurs when query executor processes for one segment use a disproportionate amount of CPU compared to processes for other segments executing the query. This value is calculated as

$$1 - (\text{average_segment_CPU} / \text{maximum_segment_CPU})$$

Memory

Memory consumed by all segment processes executing the query.

Spill Files

The total size of spill files created for the query. Greenplum Database creates spill files when there is insufficient memory to execute the query in memory. See [Managing Spill Files Generated by Queries](#) for information about spill files.

Disk R

The average disk read rate for all segment hosts.

Disk W

The average disk write rate for all segment hosts.

Locks and Blocks

Contains lists of queries blocked by locks this query holds, and queries held by other transactions that block this transaction. Click a list to expand and view the contents.

Query Text and Execution Plan

The query text and the query's plan and execution progress are shown in the lower panels of the Query Details view. The text of the query is displayed in the left panel, and the plan and progress is displayed in the right panel. The plan is available only for queries that ran for at least ten seconds.

Query Text

The **Query Text** panel displays the text of the query as it was submitted to Greenplum Database.

Command Center can display up to 100K characters. If you click **COPY**, up to 100K characters of the query text are copied to the clipboard.

If the query text is longer than 100K characters, a message is displayed with a link you can use to download the full text of the query. The name of the text file is the ID of the query with a `.txt` extension. The file is available to download for 24 hours.

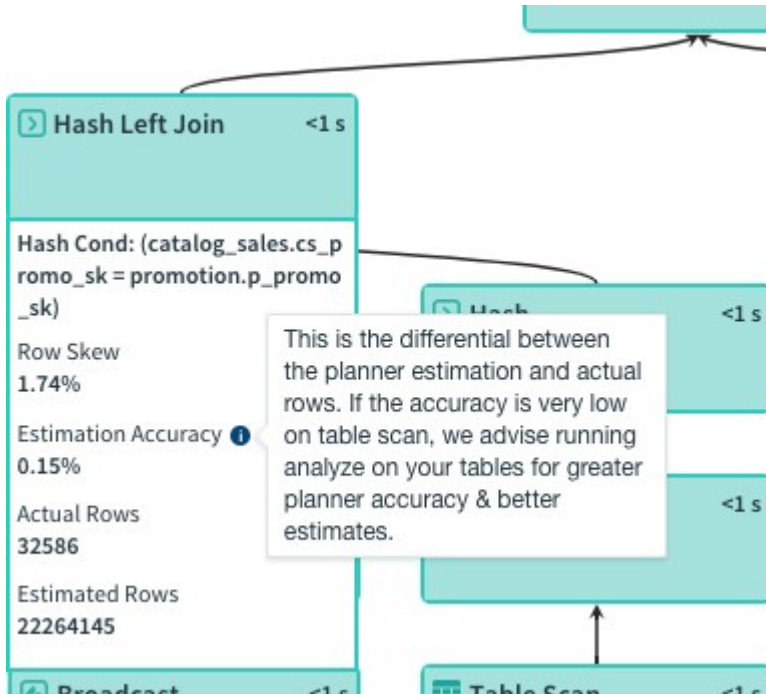
Query Plan and Progress

The **Plan & Progress** tab in the lower right panel is a graphical representation of the query plan with metrics to show the planned and actual query execution. Each box in the tree represents a step in the query execution plan. The boxes are labeled with the operation they represent. Query execution begins at the bottom of the tree and ends at the top.

Query execution plans are executed in "slices," portions of the query plan that segments can work

on independently in parallel. The plan is sliced wherever a data motion operator occurs. The time displayed in the upper right corner of each box is the amount of CPU time used for all steps in the slice to which the step belongs. The visual query plan does not illustrate slices, but you can find information about slices in the textual plan.

If you click a step, the box expands to show additional details.



The expanded plan box contains the following metrics.

- The type of operator. When the operator is a table scan, the name of the scanned table is included. See [Query Plan Execution](#) for descriptions of the operators.
- Information related to the current operation, such as the hash key, merge key, join condition, or filter condition.
- Row Skew - the amount of row skew for the current operator, a value from 0% to 100%. Row skew occurs when some segments process more rows than other segments. The percentage is calculated as $(1 - (\text{average_segment_rows} / \text{maximum_segment_rows})) * 100$.
- Estimation Accuracy - a percentage calculated from the estimated rows the node would produce and the actual rows the node produced when the query executed. The percentage is calculated as $\min(\text{estimated_rows}, \text{actual_rows}) / \max(\text{estimated_rows}, \text{actual_rows}) * 100$
- Actual Rows - The number of rows produced by this step.
- Estimated Rows - The estimated number of rows the operator will produce.

Textual Plan

Select the **Textual Plan** tab and click **RUN EXPLAIN** to generate the text representation of the explain plan. The `EXPLAIN` command is run against the current state of the database, so the plan generated may differ from the the plan used when the query executed.

The **RUN EXPLAIN** button is dimmed if Command Center is unable to generate the explain plan. Command Center is unable to generate the explain plan if the size of the query text is greater than

100K characters or if the query text contains multiple statements.

Query ID: 1527709348-39-3 ● Running Run Time 4m 9s CANCEL QUERY Current Time 2018-05-30 14:47:53 help

Details	Current Performance		Blocking (0)	Blocked by (0)
User: tpch_4	CPU: 11.08 %	CPU Skew: 41.67 %		
Database: gpadmin	Memory: 511.64 MB	Spill Files: 608.69 MB		
Res Queue: pg_default	Disk R: 0.00 MB/s	Disk W: 2.06 MB/s		
Submitted: 14:43:51				
Queued Time: 0s				
Run Time: 4m 9s				

Query Text

```

select
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice,
  sum(l_quantity)
from
  customer,
  orders,
  lineitem
where
  o_orderkey in (
    select
      l_orderkey
    from
      lineitem
    group by
      l_orderkey having
        sum(l_quantity) > 5
  )
  and c_custkey = o_custkey
  and o_orderkey = l_orderkey
group by
  c_name,
  c_custkey,
  o_orderkey,
  o_orderdate,
  o_totalprice
order by
  o_totalprice desc,
  o_orderdate
;
                
```

Plan & Progress Textual Plan

```

Gather Motion 4:1 (slice3; segments: 4) (cost=0.00..6554.33 rows=2402130 width=49)
Merge Key: orders.o_totalprice, orders.o_orderdate
-> GroupAggregate (cost=0.00..6160.61 rows=600533 width=49)
  Group By: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
  -> Sort (cost=0.00..6115.27 rows=600533 width=48)
    Sort Key: orders.o_totalprice, orders.o_orderdate, customer.c_name, customer.c_custkey, orders.o_orderkey
    -> Hash Join (cost=0.00..2977.88 rows=600533 width=48)
      Hash Cond: public.lineitem.l_orderkey = orders.o_orderkey
      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
      -> Hash (cost=1888.16..1888.16 rows=124319 width=41)
        -> Redistribute Motion 4:4 (slice2; segments: 4) (cost=0.00..1888.16 rows=124319 width=41)
          Hash Key: orders.o_orderkey
          -> Hash Join (cost=0.00..1872.21 rows=124319 width=41)
            Hash Cond: orders.o_custkey = customer.c_custkey
            -> Redistribute Motion 4:4 (slice1; segments: 4) (cost=0.00..1371.15 rows=124319 width=22)
              Hash Key: orders.o_custkey
              -> Hash EXISTS Join (cost=0.00..1362.59 rows=124319 width=22)
                Hash Cond: orders.o_orderkey = public.lineitem.l_orderkey
                -> Table Scan on orders (cost=0.00..459.64 rows=374619 width=22)
                -> Hash (cost=784.20..784.20 rows=124319 width=4)
                  -> Result (cost=0.00..784.20 rows=124319 width=4)
                    Filter: (sum(public.lineitem.l_quantity)) > 5::numeric
                    -> HashAggregate (cost=0.00..773.98 rows=310798 width=12)
                      Group By: public.lineitem.l_orderkey
                      -> Table Scan on lineitem (cost=0.00..558.99 rows=1501332 width=11)
                    -> Hash (cost=434.95..434.95 rows=37596 width=23)
                      -> Table Scan on customer (cost=0.00..434.95 rows=37596 width=23)
                
```

Settings: optimizer=on
Optimizer status: PQO version 2.58.0

The textual plan is the output of the Greenplum Database `EXPLAIN` command for the query. The query plan steps are labeled with arrows (->) and the structure of the query plan tree is indicated with indentation.

The `Optimizer status:` line at the bottom of the textual plan reports whether the explain plan was generated using the GPORCA optimizer (PQO) or the legacy query optimizer.

For help reading the textual explain plan see the `EXPLAIN` command in the *Greenplum Database Reference Guide* and *Query Profiling* in the *Greenplum Database Administrator Guide*. See *Query Execution* for descriptions of the query operators.

Alerts

On the **Admin > Alerts** page, an administrator can set up **alert rules** to detect and respond to events occurring in the Greenplum Database system and in currently executing database queries. When a rule is matched, Command Center logs a record.

You can set up email alerts by configuring an SMTP server in Greenplum Database or in Command Center. Additionally, you can create a `send_alert.sh` shell script to forward alerts to other destinations, such as an SMS gateway or a Slack channel. If the script is present, Command Center runs it whenever an alert is raised.

Command Center creates the `gpmetrics` schema in the `gpperfmon` database to store both rules and log records. See [gpmetrics Schema Reference](#) for information about the `gpcc_alert_rule` and `gpcc_alert_log` tables in the `gpmetrics` schema.

This topic contains the following subtopics:

- [Configuring Alert Rules](#)
- [Configuring Alert Email](#)
- [Creating a Send Alert Script](#)

Configuring Alert Rules

Click **EDIT** to manage alert event rules. To enable an alert rule, enter any data required in the fields and check the box. Uncheck the box to disable the rule. Click **SAVE** when you have finished making changes to the alert configuration.

CANCEL
SAVE

Alerts Events

Receive email alerts for selected events:

<input checked="" type="checkbox"/> Database connectivity failure	<input checked="" type="checkbox"/> Out of memory errors
<input checked="" type="checkbox"/> Segment failure	<input checked="" type="checkbox"/> Spill files for a query exceeds <input type="text" value="250"/> GB
<input checked="" type="checkbox"/> Average memory (segment hosts) exceeds <input type="text" value="65"/> % for <input type="text" value="10"/> min	<input type="checkbox"/> Query runtime exceeds <input type="text" value=""/> min
<input type="checkbox"/> Memory (master) exceeds <input type="text" value=""/> % for <input type="text" value=""/> min	<input checked="" type="checkbox"/> Query is blocked for <input type="text" value="15"/> min
<input type="checkbox"/> Total disk space exceeds <input type="text" value=""/> % full	
<input checked="" type="checkbox"/> Number of connections exceeds <input type="text" value="5"/>	
<input type="checkbox"/> Average CPU (segment hosts) exceeds <input type="text" value=""/> % for <input type="text" value=""/> min	
<input type="checkbox"/> CPU (master) exceeds <input type="text" value=""/> % for <input type="text" value=""/> min	

Database Connectivity Failure

An alert is raised when either of these conditions is detected:

- Command Center is running on the master host, the master host is up, but the database is down or cannot be reached.

- Command Center is running on the standby master host, the standby master host is up, but the master host is down or cannot be reached, or the database is down or cannot be reached.
- Command Center checks three times before raising the alert. If the host where Command Center should be running is down, no alert is raised.

Segment failure

An alert is raised when one or more failed segments are detected. After the alert email is raised, Command Center will raise the alert every 30 minutes until the segments are recovered.

Average memory (segment hosts) exceeds [%] for [N] min

An alert is raised when the average memory for all segment hosts exceeds the specified percentage for the specified number of minutes. Command Center samples all segment hosts every 15 seconds and calculates the mean of the samples. Only memory in use is considered; memory for buffers and cache is not included.

Memory (master) exceeds [%] for [N] min

An alert is raised when the percent of memory used on the master host exceeds the specified percentage for the specified number of minutes. Command Center samples memory usage on the master host every 15 seconds and calculates the mean of the samples. Only memory in use is considered; memory for buffers and cache is not included.

Total disk space exceeds [%] full

An alert is raised when the total of disk space in use for all segment hosts exceeds the specified percentage. Command Center gathers the *available disk space* and *total disk space* from each segment host in the Greenplum Database cluster. The *percent of total disk space in use* is calculated by the following formula:

$$100 - \text{sum}(\langle \text{available disk space} \rangle) / \text{sum}(\langle \text{total disk space} \rangle) * 100$$

A disk space alert is raised no more than once every 24 hours.

Number of connections exceeds [N]

An alert is raised when the total number of database connections exceeds the number specified. The number of connections is checked every 30 seconds. After an alert is raised, the metrics collector checks the number of connections every 30 minutes until the number of connections drops below the threshold, and then it resumes checking every 30 seconds.

Average CPU (segment hosts) exceeds [%] for [N] min

An alert is raised when the average percent of CPU used for all segment hosts exceeds the specified percentage for the specified number of minutes. Command Center samples all segment hosts every 15 seconds and calculates the mean of the samples.

CPU (master) exceeds [%] for [N] min

An alert is raised when the CPU usage on the master host exceeds the specified percentage for the specified number of minutes. Command Center samples CPU usage on the master host every 15 seconds and calculates the mean of the samples.

Out of memory errors

An alert is raised when an executing query fails with an out of memory (OOM) error. Note that no alert is raised if there is insufficient memory to start the query.

Spill files for a query exceeds [GB]

An alert is raised when the total disk space consumed by a running query's spill files exceeds the specified number of gigabytes. An alert is raised only once per query.

Query runtime exceeds [N] min

An alert is raised when a query runtime exceeds the number of minutes specified. This alert is raised just once for a query.

Query is blocked for [N] min

An alert is raised if a query remains in a blocked state for longer than the specified number of minutes. If an alert is raised, and then the query unblocks, runs, and blocks again for the specified time, an additional alert is raised. Blocked time excludes the time a query is queued before it runs. It is possible for a "Query runtime exceeds [N] min" rule to also trigger while a query is blocked.

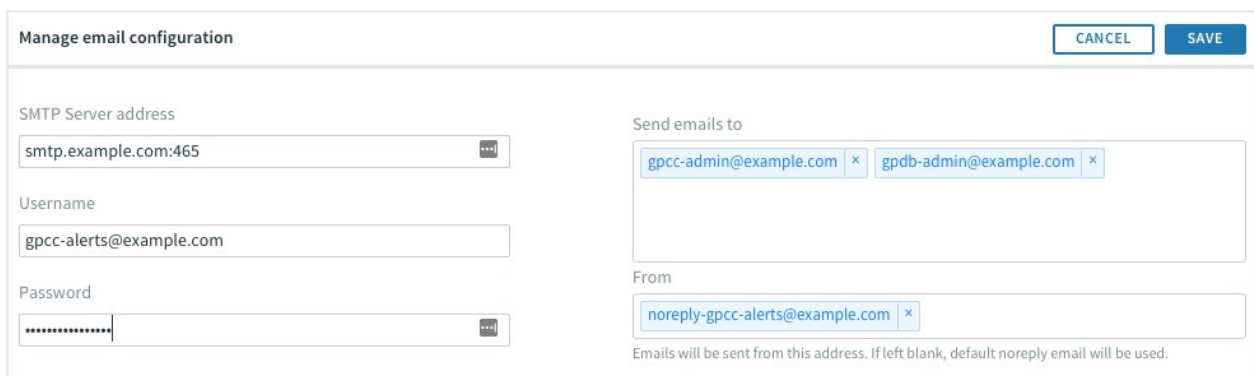
Configuring Alert Email

Command Center requires an SMTP server to send alert emails.

Command Center first attempts an encrypted TLS connection and then falls back to an unencrypted connection if TLS is not supported. The SMTP server must support one of the following authentication methods: NONE, PLAIN, LOGIN, or CRAM-MD5. Command Center will use the most secure of these methods the SMTP server allows.

Configuring email With Command Center

Click **EDIT** in the **Manage email configuration** panel.



The alert email configuration is set with the following Greenplum Database server configuration parameters:

SMTP Server address

The name or IP address of the SMTP server and the SMTP port number. The port number is typically 587 for connections with TLS encryption or 465 without encryption. Example:

```
smtp.example.com:465
```

Username

The username of the account to authenticate with the SMTP server. This is an optional field, only required if the SMTP server requires a username for authentication. Example: `gpcc-alerts@example.com`

Password

The password for the SMTP username. For security, the password is masked. This field is optional, only needed if the SMTP server requires a username and password for authentication.

Send emails to

To add an address to the list, enter the address and press Enter. To remove an email address, click the `x` on the address.

From

The email address to use for the **From:** address in the alert email. Example: `do-not-reply@example.com`. If you leave this field blank, Command Center uses the default value, `noreply-gpcc-alerts@pivotal.io`.

When you click **SAVE**, Command Center sends a test email to the addresses in the **Send emails to** field. The email contains a list of the currently configured alert rules. If there is an error in the SMTP server or username/password configuration and the email cannot be sent, Command Center displays an error message.

Configuring email for Greenplum Database

The following server configuration parameters are used to configure SMTP email for Greenplum Database.

`gp_email_smtp_server`

The SMTP server and port. Example: `smtp.example.com:465`

`gp_email_smtp_userid`

The name of a user to authenticate with the SMTP service. Example: `gpcc-alerts@example.com`

`gp_email_smtp_password`

The password for the SMTP user.

`gp_email_from`

The email address to set as the email sender. Example: `noreply-gpcc-alerts@example.com`

`gp_email_to`

A semicolon-separated list of email addresses to receive alert messages. Example `gpcc-admin@example.com;gpdb-admin@example.com`

Command Center uses the `gp_email_smtp_server`, `gp_email_smtp_userid`, and `gp_email_smtp_password` parameters if they are set. It ignores the remaining parameters.

You can check the current value of a configuration parameter by running the `gpconfig -s` command on the master host, for example:

```
$ gpconfig -s gp_email_smtp_server
```

Use the `gpconfig -c` option to set the values of server configuration parameters, for example:

```
$ gpconfig -c gp_email_smtp_server -v "smtp.example.com:465"
$ gpconfig -c gp_email_smtp_userid -v "gpcc-alerts@example.com"
$ gpconfig -c gp_email_smtp_password -v "changeme"
$ gpconfig -c gp_email_from -v "gpcc-alerts@example.com"
$ gpconfig -c gp_email_to -v "gpcc-admin@example.com;gpdb-admin@example.com"
```

Run `gpstop -u` to reload the configuration files after changing these configuration parameters.

Creating a Send Alert Script

The send alert script is a shell script that you can use to send Command Center alerts to destinations such as SMS gateways, pagers, team collaboration tools like Slack, chat servers, archive files, alternative email servers, and so on. You can use the send alert script in addition to sending email from Command Center, or as an alternative to sending alert emails from Command Center.

Command Center looks for the script `$MASTER_DATA_DIRECTORY/gpmetrics/send_alert.sh` on the host where Command Center is running—either the master host or standby host. If the file exists and is executable by the `gpadmin` user, Command Center executes the script. The following variables are set on the command line when the script runs.

Variable	Description
LINK	URL of the Greenplum Command Center web server.
QUERYID	ID of the query, if the alert was triggered by a query.
SERVERNAME	Name of the Greenplum Command Center server.
QUERYTEXT	The text of the query, if the alert was triggered by a query.
ACTIVERULENAME	Current text the of rule, with user-specified values included.
LOGID	Value of this alert's <code>id</code> column in the <code>gpmetrics.gpcc_alert_log</code> table.
RULEDESCRIPTION	Text of the rule, including user-specified values, at the time the alert was raised.
ALERTDATE	Date the alert was raised.
ALERTTIME	Time the alert was raised.
SUBJECT	Subject line for email.

An example script that you can customize is provided at `$GPCC_HOME/alert-email/send_alert.sh.sample`. The example formats the alert as HTML email text and pipes it through the Linux `mail` command.

To set up a send alert script:

1. Copy the `$GPCC_HOME/alert-email/send_alert.sh.sample` file to `$MASTER_DATA_DIRECTORY/gpmetrics/send_alert.sh`.
2. Customize the script with code to format and deliver the alert to your desired destination.

3. Run `gpcc start` to restart Command Center and enable the script.

Managing Greenplum Database Workloads

[About Workloads](#)

[Managing Greenplum Database Resource Groups](#)

[Importing Resource Queues to Resource Groups](#)

[Accessing the Workload Management Configuration Programmatically](#)

[Troubleshooting Enabling Resource Groups](#)

About Greenplum Command Center Workload Management

Greenplum Database segment hosts have a set amount of memory, CPU, I/O, and network resources. If these resources start to run short due to heavy database load, queries can fail or the entire database system can become unacceptably slow. For this reason, it is important to manage database requests to prevent resource depletion.

Greenplum Database has two resource management systems: *resource queues* and *resource groups*. Command Center workload management is based on resource groups, the resource management system introduced in Greenplum Database version 5. Resource groups require enabling Linux control groups (cgroups), so Greenplum Database initially is set to perform resource management using resource queues.

When Command Center starts, it checks the resource management system enabled in your Greenplum Database system. If you have not yet enabled resource groups in your Greenplum Database system, the Command Center workload management view displays a message encouraging you to enable resource groups, including a link to documentation with the procedure for completing the task. When you start Command Center after enabling resource groups, click the **ENABLE WORKLOADS** button. Command Center presents a view to help you set the initial resource group configuration by importing your existing resource queues to resource groups.

See [Using Resource Groups](#) in the *Greenplum Database Administrator Guide* for a full description of resource management features available with resource groups and instructions to enable resource groups in Greenplum Database.

On the Command Center **Admin> Workload Mgmt** view, you can perform the following tasks:

- Create new resource groups in Greenplum Database
- Delete existing resource groups
- Change the number of concurrent transactions each resource group allows
- Change the percentages of available system CPU and memory each resource group manages
- Change Greenplum Database roles' default resource groups

- Write query assignment rules to override a user's default resource group when a transaction executes

For more information about Linux cgroups and Greenplum Database resource groups see [Using Resource Groups](#) in the *Greenplum Database Administrator Guide*.

About Query Assignment Rules

Greenplum Database defers to the metrics collector database extension to assign transactions to resource groups. Command Center users with Admin permission level can create assignment rules in Command Center to assign transactions to resource groups.

When a transaction begins, Greenplum Database calls the metrics collector extension to determine the resource group. The extension evaluates the assignment rules and, if a matching rule is found, returns that rule's resource group. If no assignment rule matches, Greenplum Database falls back to the default behavior, assigning the transaction to the resource group specified for the current user in the `pg_authid` system table.

Assignment rules can redirect a transaction to a resource group based on query tags or a combination of query tags and the Greenplum Database role executing the transaction.

A query tag is a user-defined `name=value` pair that you can set in a Greenplum Database session when resource group-based workload management is enabled. Query tags are defined by setting the `gpcc.query_tags` parameter on the connect string when requesting a database connection or in the session with `SET gpcc.query_tags TO '<query-tags>'`. Multiple query tags can be set by separating them with a semicolon. Query tags are set before a transaction begins and cannot be changed inside of a transaction.

See [Workload Management](#) for details about creating assignment rules and for examples that use query tags.

Workload Management

Defining Resource Groups and Resource Attributes

Command Center allows you to view resource groups that have been created in Greenplum Database, to add or delete resource groups, and to edit the resource group attributes **Concurrency**, **CPU %**, and **Memory %**.

To change values of the `MEMORY_AUDITOR`, `CPUSET`, `MEMORY_SHARED_QUOTA`, or `MEMORY_SPILL_RATIO` resource group attributes, use the `ALTER RESOURCE GROUP` SQL command.

Resource Groups					EDIT
Name	Concurrency	CPU %	Memory %	Min Mem per query	
default_group	10	10	10	57.44MB	
admin_group	5	10	10	114.88MB	
etl	10	10	10	91.90MB	
priority_low	12	20	12	91.90MB	
priority_high	25	30	40	147.05MB	

1. Click **EDIT** to open the Resource Group editor.
2. To delete a resource group, select the resource group, and click the minus sign that appears at the right.

You cannot delete the `default_group` or `admin_group` resource groups. You cannot delete a resource group that is assigned to any Greenplum Database role.

3. To add a resource group, click **ADD RESOURCE GROUP** and enter a name for the resource group in the **Name** column. Resource group names must be unique and are case-sensitive.
4. Adjust the values of the **Concurrency**, **CPU %**, and **Memory %** resource group attributes.

Concurrency

The maximum number of concurrent transactions, including active and idle transactions, that are permitted in the resource group. **Concurrency** sets the `CONCURRENCY` attribute of the resource group. The total of the **Concurrency** columns cannot exceed the value of the Greenplum Database `max_connections` master server configuration parameter.

CPU %

The percentage of CPU resources available to this resource group. The percentage is the portion of the total CPU percentage allocated for all resource groups (reserved CPUs excluded), which is set with the `gp_resource_group_cpu_limit` server configuration parameter. **CPU %** sets the `CPU_RATE_LIMIT` attribute of the resource group.

Memory %

The percentage of memory resources available to this resource group. The percentage is the portion of the total memory allocated for all resource groups, which is set with the `gp_resource_group_memory_limit` Greenplum Database configuration parameter. Changing the **Memory %** value sets the `MEMORY_LIMIT` attribute of the resource group.

Min memory per query

The minimum amount of memory allocated to a query. This column is recalculated as you adjust **Concurrency** and **Memory %** settings. The value is the resource group's total share of system memory, less the resource group's shared memory pool (20% by default), divided by the value in the **Concurrency** column. The percentage of memory allocated to the shared memory pool can be changed by setting the `MEMORY_SHARED_QUOTA` attribute of the resource group using the **ALTER RESOURCE GROUP** SQL command. Each query managed by the resource queue is allocated this amount of memory. If a query needs more memory, it is allocated from the resource group shared memory pool and the global shared memory pool, if available.

The totals of the **CPU %** and **Memory %** columns must not exceed 100%. You should not allow the total of the **CPU %** column to exceed 90%, because this could cause resource-intensive queries to consume nearly all CPU, starving other Greenplum Database processes. If the total of the **Memory %** column is less than 100%, the unreserved memory is part of the resource group shared global memory pool. See "Global Shared Memory" in [Using Resource Groups](#) in the *Greenplum Database Administrator Guide* for information about the global resource group shared memory pool.

5. Click **Apply** to save your changes or click **Cancel** to abandon your changes.

Assigning Roles to Resource Groups

Every Greenplum Database role is assigned to a single resource group in the `pg_roles` system table. Transactions executed by a role are managed by its assigned resource group, unless you create an assignment rule to override the default.

You can view the current resource group assignments for all roles and change a role's resource group by adding it to a different resource group.

Assignment by Role
Queries are routed to Resource Group based on gpdb role, unless diverted by Query Tag filter(below)

default_group(3)	admin_group(2)	etl(1)	priority_low(2)	priority_high(1)
<div style="border: 1px solid #ccc; min-height: 50px;"> ralph sallyr cashk </div>	<div style="border: 1px solid #ccc; min-height: 50px;"> gpadmin gpmon </div>	<div style="border: 1px solid #ccc; min-height: 50px;"> nickd </div>	<div style="border: 1px solid #ccc; min-height: 50px;"> tpch_4 tpch_1 </div>	<div style="border: 1px solid #ccc; min-height: 50px;"> kristiem </div>
<input type="text" value="add role"/>	<input type="text" value="add role"/>	<input type="text" value="add role"/>	<input type="text" value="add role"/>	<input type="text" value="add role"/>

To move a role to a different resource group:

1. Enter all or part of the role name in the **add role** field beneath the new resource group.
2. Choose the role from the list that is displayed and press Enter.

The change is immediately applied to the Greenplum Database `pg_roles` system table.

Defining Workload Management Rules

Query assignment rules allow you assign transactions to a resource group based on user-defined query tags and, optionally, the current role in the database session. When no rule matches, the transaction is assigned to the role's default resource group. See [About Assignment Rules](#) for more information about assignment rules.

See [Accessing the Workload Configuration Programmatically](#) for information about retrieving and setting rules programmatically with database functions.

Workload Management Rules

Queries matching a filter's query tag connection attributes will be diverted, overriding its role's assigned Resource Group. Drag to change filter order.

If matching query tags		Assign to Resource Group
<input type="text" value="fdsa=2"/> AND <input type="text" value="cc_selfonly"/>		<input type="text" value="default_group"/>

1. Click **EDIT** to open the Workload Management Rules editor.
2. To delete a rule, select the rule and click the minus sign that appears at the right.
3. To add an assignment rule, click **ADD ASSIGNMENT RULE** and fill in the fields.

Query Tags

The first field is a list of query tags to match against the `gpcc.query_tags` parameter in the Greenplum Database session. A query tag is a user-defined `<name>=<value>` pair.

Separate multiple query tags with semicolons. See [Defining and Setting Query Tags](#) for more information about query tags.

Role

(Optional) If you enter a role name in this field, the rule matches only if both the query tags and role match the tags and current role in the database session.

Resource Group

Choose a resource group from the list.

- ◆ Change the order of the assignment rules by dragging a rule's handle (at the left) up or down. Assignment rules are evaluated from top to bottom. Greenplum Database applies the first rule that matches.
- ◆ Use the **Active/Inactive** toggle to make a rule active or inactive.

4. Click **APPLY** to save your changes.

Defining and Setting Query Tags

A query tag is a user-defined `<name>=<value>` pair, set in the Greenplum Database `gpcc.query_tags` parameter in the Greenplum Database session. The `gpcc.query_tags` parameter is defined when the `gp_wlm` database extension is enabled in the postgres database. If you try to set query tags when the `gp_wlm` extension is not enabled, you get an unrecognized configuration parameter error. To see if the extension is enabled, run the following command.

```
$ psql postgres -c "\dx"
                                List of installed extensions
 Name | Version | Schema | Description
-----+-----+-----+-----
 gp_wlm | 0.1     | gpcc   | Greenplum Workload Manager Extension
(1 row)
```

When you submit a transaction and the `gp_wlm` extension is enabled, Greenplum Database calls the `gp_wlm` extension to determine the resource group for the transaction. The extension evaluates the current role and query tags set in the session against the rules you have defined in Command Center. If there is a match, the extension returns the rule's resource group. If there is no match, Greenplum Database assigns the transaction to the role's default resource group.

The following command, executed in the Greenplum Database session, sets the `appName` and `appUser` query tags to "tableau" and "bi_sales", respectively.

```
=# SET gpcc.query_tags TO 'appName=tableau;appUser=bi_sales';
```

To match a rule, all tags in the rule's query tag field must be present in the `gpcc.query_tags` parameter in the database session. The order of the tags is not significant, and the `gpcc.query_tags` parameter can have a superset of the tags defined in the `queryTags` value.

If you set the `gpcc.query_tags` parameter inside of a transaction, you must commit the transaction before the new query tags are used to evaluate assignment rules.

You can set the value of the `gpcc.query_tags` parameter using the `SET` command, as in the example above, or as a connection parameter with database clients that support it, such as `psql`. Following are

two examples that show how to specify query tags on the `psql` command line.

```

$ PGOPTIONS="-c gpcc.query_tags='appName=tableau;appUser=bi_sales'" psql

$ psql postgresql://mdw:5432/postgres?options="-c gpcc.query_tags%3D'appName%3Dtableau;appUser%3Dbi_sales'"
    
```

In the second example, it is necessary to code the equals signs as `%3D` to prevent `psql` from interpreting the query tags as command-line arguments.

Importing Resource Queues to Resource Groups

Greenplum Command Center workload management works with resource groups, the new Greenplum Database resource management system. The default resource management system for Greenplum Database is resource queues. To use the Command Center workload management features, you must first enable resource groups in Greenplum Database.

Command Center can assist you in enabling resource groups and in importing existing resource queues to resource groups.

Step One: Enable Resource Groups in Greenplum Database

If your Greenplum Database system is still configured to use resource queues, the Command Center **Admin> Workload Mgmt** view describes the benefits of resource groups and workload management with Command Center and provides a link to the Greenplum Database documentation to help you enable resource groups.

Workload Management Manage your workload through Resource Groups

With Resource Groups and Command Center, you can

- Handle mixed workloads**
Assign queries to Resource Groups based on role or custom query tags
- Use system resources more efficiently**
Control memory, CPU, and concurrency allocation for each Resource Group
- Support SLAs**
Protect resources for privileged groups to ensure queries complete on time
- Improve memory management**
Prevent over subscription and isolate memory between Resource Groups and transactions

[VIEW RESOURCE GROUP SET UP GUIDE](#)

Click **VIEW RESOURCE GROUP SET UP GUIDE** for instructions to enable resource groups in your Greenplum Database system.

Step Two: Preview and Configure Resource Group Imports

After you have enabled resource groups and restarted Greenplum Database, restart Command Center (`gpcc start`), log in, and choose **Admin> Workload Mgmt**.

The workload management view now displays a preview of resource groups converted from your

existing resource queues. You can use this one-time view to convert your Greenplum Database resource queues to resource groups.

Workload Management Manage your workload through Resource Groups

Here is a preview of your resource groups converted from your resource queues. Please input the resource allocations. Your roles will be matched with the assigned resource groups.

Resource Group	Concurrency	CPU %	Memory %	Min mem per query
default_group	<input type="text" value="20"/>	<input type="text" value="30"/>	<input type="text" value="30"/>	86.16MB
admin_group	<input type="text" value="20"/>	<input type="text" value="30"/>	<input type="text" value="30"/>	86.16MB
vip	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
etl	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
adhoc	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
bi_analytics	<input type="text"/>	<input type="text"/>	<input type="text"/>	--
		60	60	Total CPU and Memory must be less than or equal to 100%

IMPORT RESOURCE GROUPS **SKIP IMPORT**

Your roles will be imported to the matching resource groups. [Close preview.](#)

default_group (0)	admin_group (2)	vip (3)	etl (2)	adhoc (0)	bi_analytics (4)
	gpmon gpadmin	nickd ralphs katrinab	kristiem richd		jillianr brentd sallyg anny

The resource group list includes the required `admin_group` and `default_group` resource groups, and a row for each of your existing resource queues.

Roles are assigned to the resource group matching the resource queue to which they are assigned. Click the **Preview roles** link to see the role assignments.

Your roles will be imported to the matching resource groups. [Close preview.](#)

default_group (0)	admin_group (2)	vip (3)	etl (2)	adhoc (0)	bi_analytics (4)
	gpmon gpadmin	nickd ralphs katrinab	kristiem richd		jillianr brentd sallyg anny

If you want to set up resource groups later, you can click **SKIP IMPORT**. Only the `default_group` and `admin_group` resource groups are created. Roles with the superuser attribute are assigned to the `admin_group` resource group; roles without superuser privilege are assigned to the `default_group` resource group.

If you want Command Center to import resource queues to resource groups, you must complete the resource allocation fields for all resource groups.

Set the **Concurrency**, **CPU %**, and **Memory %** resource group attributes to allocate Greenplum Database resources to the resource queues. The **Concurrency** fields must each contain a positive integer. The **CPU %** and **Memory %** fields must each contain positive integers between 1 and 99 and the totals for the **CPU %** and **Memory %** columns must not exceed 100%. See [Defining Resource Groups and Resource Attributes](#) for help determining the values to enter.

The **IMPORT RESOURCE GROUPS** button is disabled until you have entered valid values in the allocation fields for every resource group.

When you are ready to import the resource groups, click **IMPORT RESOURCE GROUPS** to create the resource groups.

Step Three: Enable Command Center Workload Management

After you import (or skip importing) resource queues to resource groups, you can enable Command Center workload management.

Name	Concurrency	CPU %	Memory %	Min mem per query
default_group	5	2	5	86.16MB
admin_group	5	2	5	57.44MB
vip	10	10	15	137.86MB
etl	5	5	5	91.90MB
adhoc	5	2	3	55.14MB
bi_analytics	50	30	30	55.14MB

Assignment by Role

Queries are routed to Resource Group based on gpdb role, unless diverted by Query Tag filter (below)

Resource Group	Assigned Roles
default_group (0)	
admin_group (2)	gpmon, gpadmin
vip (3)	nickd, ralphs, katrinab
etl (2)	kristiem, richd
adhoc (0)	
bi_analytics (4)	sallyg, brentd, jillianr, anny

Workload Management Rules

Queries matching a filter's query tag connection attributes will be diverted, overriding its role's assigned Resource Group. Drag to change filter order.

If matching query tags: Assign to Resource Group

no assignment rules

ENABLE WORKLOAD MANAGEMENT

Automatically terminate connections when idle for a specified time period

Enable Workload Management will allow you to set query tags and idle session rules.

Resource Group	Time before idle connections killed	Exempted roles	Message
no timeout rules			

Click **ENABLE WORKLOAD MANAGEMENT** to enable workload management in Greenplum Command Center. Greenplum Command Center creates the `gp_wlm` extension, the `gpcc.workload_config` table, and the associated user-defined functions in Greenplum Database.

You are now able to use the Command Center Workload Management interface to add, remove, and configure resource groups; change role assignments; and define workload management rules.

See [Workload Management](#) for help using the Command Center Workload Management view.

Accessing the Workload Configuration Programmatically

The Greenplum Database workload management extension `gp_wlm` creates a table in the `gpperfmon` database to store the workload management rules, and user-defined functions to get or set the workload management rules.

The `gpmetrics.workload_config` table stores the workload management rules as a JSON value. You

can use the `gpmetrics.get_workload_config()` and `gpmetrics.set_workload_config()` functions to read and write this JSON value.

This topic is a reference for the workload management configuration JSON document and the get and set functions.

Warning! The `gpmetrics.workload_config` table should only be accessed by using the `gpmetrics.get_workload_config()` and `gpmetrics.set_workload_config()` functions or the Command Center user interface. Do not drop the table while the workload management extension is enabled.

The `gpmetrics.set_workload_config()` function requires valid JSON syntax, but does not validate the workload management rules. You must ensure that the JSON value contains a `version` variable and correctly specified assignment rules.

Workload Management Rules JSON Format

This section describes the JSON object that stores the resource group assignment rules. The object has two members:

- a `version` key/value pair
- an `assignmentRules` array containing one element for each assignment rule

version pair

version

The `version` value is an integer. It is reserved for future use. It can be set to 1.

assignmentRules array

assignmentRules

The `assignmentRules` array has one element for each assignment rule. Each element maps the rule to a resource group in Greenplum Database and defines the conditions that assign a transaction to that resource group.

Greenplum Command Center evaluates each rule in the `assignmentRules` array from top to bottom and stops at the first match, so the order is important.

The elements in the `assignmentRules` array can have the following key/value pairs.

resourceGroupName

The name of the Greenplum Database resource group. The resource group must already exist or have been created in Greenplum Database with the `CREATE RESOURCE GROUP` SQL statement.

roleName

The name of a Greenplum Database role to match against the current role identifier in the Greenplum Database session. The current role is initially the database role that authenticated with the Greenplum Database system to begin the session. A user with sufficient privileges can change the current role in the database session using the `SET ROLE` SQL command.

If no role is specified in the assignment rule and the query tags match, the transaction is assigned to the specified resource group. If the `roleName` value is present, however, the

current database user must match the specified role.

queryTags

A list of query tags to match against the `gpcc.query_tags` parameter in the Greenplum Database session. A query tag is a user-defined `<name>=<value>` pair. Separate multiple query tags with semicolons. For example, the following statement, executed in the Greenplum Database session, sets the `appName` and `appUser` query tags to “tableau” and “bi_sales”, respectively.

```
=# SET gpcc.query_tags TO 'appName=tableau;appUser=bi_sales';
```

To match, all tags in the assignment rule’s `queryTags` value must be present in the `gpcc.query_tags` parameter in the database session. The order of the tags is not significant, and the `gpcc.query_tags` parameter can be a superset of the tags defined in the `queryTags` value.

If the `queryTags` value is empty, or the parameter omitted, the rule will match every session for the `roleName` database role.

If you set the `gpcc.query_tags` parameter inside of a transaction, you must commit the transaction before the new query tags are used to evaluate assignment rules.

The `gpcc.query_tags` parameter can be specified using the `SET` command, as above, or as a connection parameter with database clients that support it, such as `psql`. Here are two examples that show how to specify query tags on the `psql` command line:

```
$ PGOPTIONS="-c gpcc.query_tags='appName=tableau;appUser=bi_sales'" psql
$ psql postgresql://mdw:5432/postgres?options="-c gpcc.query_tags%3D'appName%3Dtableau;appUser%3Dbi_sales'"
```

In the second example, it is necessary to code the equals signs as `%3D` to prevent `psql` from interpreting the query tags as command-line arguments.

disabled

If set to `true`, the assignment rule is ignored when Command Center evaluates rules to assign transactions to workloads. This parameter is optional and its default value is `"false"`.

gpmetrics.get_workload_config()

Retrieves the current workload assignment rules as a JSON value.

Example

```
gpperfmon=# SELECT gpmetrics.get_workload_config();
get_workload_config
-----
{
  "version":1,
  "assignmentRules":[
    {
      "resourceGroupName":"admin_group",
      "roleName":"optionalRoleToFilterWith",
```

```

    "queryTags": "exampleKey1=exampleValue1;exampleKey2=exampleValue2",
    "disabled": true
  },
  {
    "resourceGroupName": "default_group",
    "queryTags": "exampleKey1=exampleValue1;exampleKey2=exampleValue2",
    "disabled": true
  }
]
}
(1 row)

```

gpmetrics.set_workload_config()

Sets the workload assignment rules. The argument is a valid JSON value containing the assignment rules. See [JSON Parameters](#) for descriptions of the parameters.

The `gpmetrics.set_workload_config()` function accepts any valid JSON value. You must ensure that the value contains a `version` element and a properly specified assignment rules parameter.

If you call `gpmetrics.set_workload_config()` within a transaction, you must commit the transaction before the workload management extension applies the new rules.

Example

```

postgres=# SELECT gpmetrics.set_workload_config(
'
  { "version": 1,
    "assignmentRules":
      [
        {
          "resourceGroupName": "default_group",
          "roleName": "gpadmin",
          "queryTags": "appName=tableau;appUser=bi_sales"
        },
        {
          "resourceGroupName": "admin_group",
          "roleName": "gpadmin",
          "queryTags": "appName=tableau;appUser=bi_acct",
          "disabled": true
        }
      ]
    }
);
 set_workload_config
-----
 t
(1 row)

```

Troubleshooting Enabling Resource Groups

If you experience problems enabling resource groups in Greenplum Command Center, review the following list to ensure prerequisites are met and all of the dependencies are properly configured.

- Red Hat 6.x and 7.x and CentOS 6.x and 7.x are currently supported.
- You must be running Greenplum Database version 5.7.0 or later.
- Configure the Linux cgroups kernel feature on your hosts by following the instructions at “Prerequisite” in [Using Resource Groups](#).
- Make sure the `/etc/cgconfig.d/gpdb.conf` file contains the objects perm, cpu, and cpuacct. If the document is incorrect and the `gp_resource_manager` configuration parameter is set to `"group"`, Greenplum Database can hang at startup.

```
group gpdb {
  perm {
    task {
      uid = gpadmin;
      gid = gpadmin;
    }
    admin {
      uid = gpadmin;
      gid = gpadmin;
    }
  }
  cpu {
  }
  cpuacct {
  }
}
```

- On Red Hat 7, make sure you run `cgconfigparser -L /etc/cgconfig.d` to parse changes to the `/etc/cgconfig.d/gpdb.conf` file. This command must also be set up to run at boot time.
- Set the Greenplum Database `gp_resource_manager` server configuration parameter to `"group"` and restart Greenplum Database.

```
$ gpconfig -c gp_resource_manager -v "group"
$ gpstop -ar
```

Verify by showing the value of the parameter:

```
$ gpconfig -s gp_resource_manager
Values on all segments are consistent
GUC           : gp_resource_manager
Master value: group
Segment value: group
```

- After installing a Pivotal Greenplum Database distribution, the `shared_preload_libraries` configuration parameter contains the metrics collector shared library. Make sure this library is still present:

```
$ gpconfig -s shared_preload_libraries
Values on all segments are consistent
GUC           : shared_preload_libraries
Master value: metrics_collector
Segment value: metrics_collector
```

Check that the shared library exists at `$GPHOME/lib/postgresql/metrics_collector.so`. If the

library does not exist, make sure you have installed the Pivotal Greenplum Database distribution. This extension is not available in the Greenplum Database Open Source version.

If the shared library file exists in the `$GPHOME/lib/postgresql` directory, but not in the `shared_preload_libraries` parameter, add it with the `gpconfig` command:

```
$ gpconfig -c shared_preload_libraries -v 'metrics_collector'
```

Note that adding `metrics_collector` to the `shared_preload_libraries` parameter does not enable the `metrics_collector` or `gp_wlm` extensions, but is a prerequisite for enabling them.

- The `gpmon` user must be able to connect to databases from the Command Center host. Make sure to add a `host` entry like the following in the `$MASTER_DATA_DIRECTORY/pg_hba.conf` file.

```
host all gpmon <IP_of_host>/32 md5
```

- Check whether the `gp_wlm` extension is installed. The extension is added when you click **Enable Workloads** in the Greenplum Command Center **Admin> Workload Mgmt** view.

```
$ psql gpperfmon
\dx
gpperfmon=# \dx
                                List of installed extensions
  Name          | Version | Schema      | Description
-----+-----+-----+-----
 gp_wlm         | 0.1     | gpmetrics   | Greenplum Workload Manager Extension
 metrics_collector | 1.0     | gpmetrics   | Greenplum Metrics Collector Extension
 plpgsql        | 1.0     | pg_catalog  | PL/pgSQL procedural language
(3 rows)
```

- Make sure the `gpmetrics.workload_config` table and functions are present in the `gpperfmon` database:

```
gpperfmon=# \d gpmetrics.workload_config
Table "gpmetrics.workload_config"
  Column | Type      | Modifiers
-----+-----+-----
 dist_col | integer  |
 config   | json     |
Distributed by: (dist_col)

gpperfmon=# \df gpmetrics.*
                                List of functions
 Schema | Name                                | Result data type | Argument data types | Type
-----+-----+-----+-----+-----
 gpmetrics | get_workload_config                | json              |                      |
          | normal                              |                   |                      |
 gpmetrics | metrics_collector_start_worker     | void              |                      |
          | normal                              |                   |                      |
 gpmetrics | set_workload_config                | boolean           | wlm_json_confid    |
```

```
g json | normal
(3 rows)
```

If the `gpmetrics.workload_config` table or the functions are not present, dropping and recreating the `gp_wlm` extension will create them. Note that any assignment rules saved in the `gpmetrics.workload_config` table will have to be recreated in Command Center.

```
$ psql gpperfmon
gpperfmon=# DROP EXTENSION gp_wlm;
DROP EXTENSION
gpperfmon=# CREATE EXTENSION gp_wlm;
CREATE EXTENSION
```

Query Monitor Help Topics

- [CPU](#)
- [CPU Skew](#)
- [Locks](#)
- [Query Optimization](#)
- [Memory](#)
- [Spill Files](#)

CPU

The **CPU percent** metric is the average current CPU percentage for all backend processes executing this query. The percentages for all processes running a query on each segment are averaged, and then the average of all those values is calculated to render this metric.

You can manage the percentage of CPU that queries can consume by creating workloads and specifying the maximum percent of CPU each workload can consume. That percentage is further divided among the segments running on each host and then among the concurrent queries the workload can execute.

CPU allocated to idle workloads is reallocated to active queries and reclaimed when the idle workload becomes active again. This means that the **CPU percent** value for a query can exceed limits defined for workloads and can increase and decrease as other queries start or finish.

Memory and disk I/O resources are more likely causes for degraded query performance than lack of CPU cycles. The ways to reduce CPU contention mirror the solutions for insufficient memory:

- Reduce concurrency of workloads to make more CPU available to each query.
- Reduce the number of workloads and reallocate CPU to the remaining workloads.

If CPU is not constrained and the size of spill files for some queries is very large, make sure that the `gp_workfile_compression` server configuration parameter is set to `on`. Compressing spill files reduces disk I/O, but uses CPU cycles to compress and decompress the data.

See [Using Resource Groups](#) for more about managing performance with resource groups.

If your Greenplum Database system is configured to manage resources with resource queues, see [Using Resource Queues](#).

CPU Skew

What is CPU Skew?

CPU skew occurs when the work to execute a query is not distributed evenly among the segments.

The **CPU** metric is the average of the CPU percentages used by each process executing the query. The **CPU skew** metric is a variance statistic based on the difference between the average and each segment's current **CPU** metric. The smaller the **CPU skew**, the more equally the work is distributed. The **CPU skew** metric varies between 0.0 (no skew) and 1.0. The lower the skew metric the more fully the database cluster's resources are utilized.

CPU skew is usually related to the volume of data processed by the segments while executing the query execution plan. There are two types of skew you should investigate: data skew and computational skew.

Data Skew

A high CPU skew may be an indication of data skew, where tables used by the query are distributed unevenly, so that some segments have more data to process than their peers. You can check for data skew in a table by running a query like this one:

```
=# SELECT gp_segment_id, COUNT(*) FROM <table-name> GROUP BY gp_segment_id;
```

The row count should be approximately equal for each segment. If the rows are distributed unevenly, check the distribution key for the table. A good distribution key is a column or list of columns with unique or nearly unique values, such as the table's primary key. Setting the distribution policy to **DISTRIBUTED RANDOMLY** also ensures a well-distributed table, but precludes taking advantage of performance-enhancing strategies such as co-location for tables with equivalent primary keys.

Computational Skew

High CPU skew can be the result of computational skew, which occurs during query execution. Some of the operations in the query plan can cause some segments to do more work than others. For example, joins, sorts, or aggregations on columns with low cardinality or unevenly distributed values can contribute to CPU skew by causing some segments to process many more tuples than others.

See [Distribution and Skew](#) in the *Greenplum Database Administrator Guide* and [Tuning SQL Queries](#) in the *Greenplum Database Best Practices* guide for more help finding the causes of skew.

Locks

Greenplum Command Center displays the locks currently held by queries and queries blocked by locks.

A block occurs when one query needs to acquire a lock that conflicts with a lock held by another query. If a query is blocked for a long period of time, you can investigate the blocking query and, if necessary, cancel one of the queries.

Locks can be acquired using the **LOCK TABLE** SQL statement. Some SQL commands acquire locks automatically. Following are descriptions of the lock modes, the Greenplum Database commands that acquire them, and which lock modes conflict with them.

ACCESS SHARE

Acquired by `SELECT` and `ANALYZE` commands.

Conflicts with `ACCESS EXCLUSIVE` locks.

In general, any query that only reads a table and does not modify it acquires this lock mode.

ROW SHARE

Acquired by `SELECT FOR SHARE` command.

Conflicts with `EXCLUSIVE` and `ACCESS EXCLUSIVE` locks.

A `ROW SHARE` lock is placed on the specified table and an `ACCESS SHARE` lock on any other tables referenced in the query.

ROW EXCLUSIVE

Acquired by `INSERT` and `COPY` commands.

Conflicts with `SHARE`, `SHARE ROW EXCLUSIVE`, `EXCLUSIVE`, and `ACCESS EXCLUSIVE` locks.

A `ROW EXCLUSIVE` lock is placed on the specified table and `ACCESS SHARE` locks are placed on any other referenced tables.

SHARE UPDATE EXCLUSIVE

Acquired by `VACUUM` and `VACUUM FULL`.

Conflicts with the `SHARE UPDATE EXCLUSIVE`, `SHARE`, `SHARE ROW EXCLUSIVE`, `EXCLUSIVE`, and `ACCESS EXCLUSIVE` locks.

`SHARE UPDATE EXCLUSIVE` protects a table against concurrent schema changes and `VACUUM` runs.

SHARE

Acquired by `CREATE INDEX`.

Conflicts with `ROW EXCLUSIVE`, `SHARE UPDATE EXCLUSIVE`, `SHARE ROW EXCLUSIVE`, `EXCLUSIVE`, and `ACCESS EXCLUSIVE` locks.

Protects a table against concurrent data changes.

SHARE ROW EXCLUSIVE

This lock mode is not automatically acquired by any Greenplum Database command.

Conflicts with `ROW EXCLUSIVE`, `SHARE UPDATE EXCLUSIVE`, `SHARE`, `SHARE ROW EXCLUSIVE`, `EXCLUSIVE`, and `ACCESS EXCLUSIVE` locks.

EXCLUSIVE

Acquired by `UPDATE`, `SELECT FOR UPDATE`, and `DELETE` commands in Greenplum Database.

Conflicts with `ROW SHARE`, `ROW EXCLUSIVE`, `SHARE UPDATE EXCLUSIVE`, `SHARE`, `SHARE ROW EXCLUSIVE`, `EXCLUSIVE`, and `ACCESS EXCLUSIVE` locks. This lock mode allows only concurrent `ACCESS SHARE` locks - a table can be read by another transaction while this lock is held. This is more restrictive locking than in regular PostgreSQL.

ACCESS EXCLUSIVE

Acquired by the `ALTER TABLE`, `DROP TABLE`, `TRUNCATE`, `REINDEX`, `CLUSTER`, and `VACUUM FULL` commands. Default lock mode for `LOCK TABLE` statements that do not specify a lock mode.

Also briefly acquired by `VACUUM` (without `FULL`) on append-optimized tables during processing.

Conflicts with all locks.

This lock mode guarantees that the holder is the only transaction accessing the table in any way.

For more on locks in Greenplum Database queries, see the [LOCK](#) command Reference. See also [Tuning SQL Queries](#).

Memory

The Greenplum Command Center Query Monitor reports the current total memory consumed by all processes executing a query. When there is insufficient memory available for a query to complete, the query has an error status in the query monitor and an out of memory error is logged.

If you have enabled resource groups in Greenplum Database, you can manage the amount of memory available to queries by tuning resource group parameters, and by setting Greenplum Database configuration parameters that affect resource group memory.

- For a detailed description of resource group memory management, see [Using Resource Groups](#) in the *Greenplum Database Administrator Guide*.
- If you are using resource queues, see [Memory and Resource Management with Resource Queues](#) and [Using Resource Queues](#) for ways to troubleshoot memory problems with resource queues.
- See [Tuning SQL Queries](#) for help with query optimization.

The following summary describes the resource group parameters and related Greenplum Database server configuration parameters that determine the amount of memory available to database queries and how configuration choices affect concurrency, spill file usage, and query performance.

Resource Group Memory Configuration Parameters

A resource group has parameters `CONCURRENCY`, `MEMORY_LIMIT`, `MEMORY_SHARED_QUOTA`, and `MEMORY_SPILL_RATIO`, which determine how much memory is allocated to execute a query. The `CPU_LIMIT` parameter has no effect on memory allocation. See the `CREATE RESOURCE GROUP` SQL reference for command syntax and information about these parameters.

`MEMORY_LIMIT`

This parameter sets the amount of memory the resource group manages as a percentage of the memory available to resource groups. The sum of all resource groups' `MEMORY_LIMITS` must not exceed 100. If the sum of all resource groups' `MEMORY_LIMITS` is less than 100, the remaining, unallocated memory is *global resource group shared memory*, available to queries from all resource groups on a first-come, first-served basis.

`MEMORY_SHARED_QUOTA`

A resource group divides the memory it manages into a fixed portion and a shared portion, called *resource group shared memory*. This parameter specifies the percentage of a resource group's memory that is shared. The default is 20 and the value can range from 0 to 100.

`CONCURRENCY`

This parameter limits the number of concurrent transactions a resource group allows. The fixed portion of the memory the resource group manages is divided equally among

`CONCURRENCY` transaction slots. Every transaction starts with this fixed portion of memory and, if needed, Greenplum Database uses additional memory from the resource group shared memory and global resource group shared memory.

`MEMORY_SPILL_RATIO`

This parameter sets a limit for the amount of memory a query can use before it spills to disk. The parameter value is expressed as a percentage of the fixed memory allocation. The default is 20 and the value can range from 0 to 100. A higher value uses more memory, but can improve query performance. A transaction can override this value by setting the `memory_spill_ratio` configuration parameter in the session.

When a query executes, Greenplum Database allocates memory to it from the fixed portion of the resource group's memory. If the query needs more memory and the resource group has available shared memory, Greenplum Database allocates additional memory. If insufficient shared memory is available, Greenplum Database allocates additional memory from global shared memory, if available. If the required memory is not available the transaction fails.

Greenplum Database Memory Configuration Parameters

The following Greenplum Database configuration parameters affect resource group memory allocation and concurrency.

`gp_resource_group_memory_limit`

This Greenplum Database server configuration parameter sets the percentage of each host's system memory to be managed by resource groups. The default is 0.7 (70%). This memory is divided equally among the primary segments on each host, and further divided among resource groups with the `MEMORY_LIMIT` resource group parameter. Any memory not allocated to resource groups becomes global shared memory available to queries from all resource groups. See `gp_resource_group_memory_limit` for a complete reference for this parameter.

`gp_resgroup_memory_policy`

This parameter determines how Greenplum Database allocates memory to query operators. The default value, `eager_free`, re-allocates memory from completed operators to operators later in the query plan. The alternative value for this parameter, `auto`, allocates a fixed amount of memory to operators that are not memory-intensive and the rest to the memory-intensive operators. The default value is usually the optimal setting. See `gp_resgroup_memory_policy` for a complete reference for this parameter.

`memory_spill_ratio`

A transaction can override the resource group's `MEMORY_SPILL_RATIO` value by setting the `memory_spill_ratio` configuration parameter in the session. The value is a percentage of the fixed memory allocation for transactions in the resource group, expressed as an integer from 0 to 100. The performance of queries with low memory requirements can be improved by setting the `memory_spill_ratio` parameter in the session to a low setting, for example 0 to 2. See `memory_spill_ratio` for more information about this parameter.

Query Plan Execution

The Greenplum Database legacy and GPORCA query optimizers generate execution plans that produce the results requested by the query. A plan is a sequence of operators, such as table scans,

joins, sorts, aggregates, and data motions.

When you select a query on the Command Center **Query Monitor** view, a **Query Details** view presents a graphical representation of the execution plan.

You can switch between the graphical and textual representations of the query execution plan by selecting the **Plan & Progress** tab or the **Textual Plan** tab. In the textual format, each plan node is flagged with an arrow (->). In the graphical view, the nodes are represented by boxes that fill as the plan executes.

A query execution plan executes from the bottom up. Each node in the plan performs an operation and passes results up to the next node in the plan.

The **Optimizer status:** line on the **Textual Plan** tab reports whether the explain plan was generated using the GPORCA optimizer or the legacy query optimizer.

Slices and Gangs

Segments can work on portions of a query in parallel, each segment executing operators independently on their local data. When the plan requires exchanging data between segments, a data motion operator coordinates the data transfer between segments. The plan is divided into “slices” where these data motions occur.

A data motion node in a textual query plan identifies the slice and the number of segments participating in the motion.

Example:

```
-> Broadcast Motion 4:4 (slice2; segments: 4) (cost=0.00..867.15 rows=10000 width=30)
```

In a broadcast motion, each segment broadcasts all of its rows for a table over the network so that every segment has a complete copy of the table. In this example, the broadcast motion marks the completion of `slice2` with four segments sending and four segments receiving.

Each segment has one or more backend processes working on a slice. Backend processes working on the same slice are called a “gang”.

Operators

Operators are processes that take as input database tables or the output from other operators, and perform some action to produce a transformed output.

Scan Operators

Init plan

A query that runs before the main query is optimized to find the partitions to scan.

Sequential scan

The optimizer may choose a sequential table scan if there is no index on the condition column or if most rows are expected to satisfy the condition. Because each segment scans an equal portion of the data in parallel with other segments, a table scan in Greenplum Database is very efficient. A query on a partitioned table may be able to eliminate partitions to make the scan

even faster.

Append-only scan

Scans rows in a row-oriented, append-optimized table.

Append-only columnar scan

Scans rows in a column-oriented, append-optimized table.

Dynamic table scan

Scans selected partitions in a partitioned table.

Function scan

A Function Scan node selects the partitions to scan. The function can be one of the following:

- `gp_partition_expansion` - chooses all nodes
- `gp_partition_selection` - chooses a partition with an equality expression
- `gp_partition_inversion` - chooses partitions with a range expression

Index scan

Scans a B-tree index on a table to find rows. The rows are then retrieved from disk.

Bitmap index scan

A Bitmap Index Scan is an index scan optimized by storing rows in a bitmap instead of retrieving them from the table immediately. When the scan is complete, rows in the bitmap are retrieved with a Bitmap Heap Scan operation.

BitmapAnd and BitmapOr

Generates a new bitmap by running logical AND or OR on multiple bitmaps.

Bitmap heap scan

Retrieves rows from heap storage using a bitmap generated by a Bitmap index scan or BitmapAnd or BitmapOr operation.

Nested loop with inner sequential scan join

For each row in the first table, the operator tests every row in the second table with a sequential scan.

One table must be broadcast so that every segment can compare all rows of one table with the rows it has from the other table. This is expensive and is best used only for small tables.

Nested loop with inner index scan

For each row in the first table, the operator searches an index on the second table.

One table must be broadcast so that every segment can compare all rows of one table with the rows it has from the other table.

Append

Concatenates data sets. For example, combines rows scanned from multiple partitions.

Filter

Selects rows using a `WHERE` clause.

Limit

Limits the number of rows returned.

Materialize

Saves results from a subselect so that it is not necessary to process the inner table for every

row in the outer table.

Join Operators

Hash join

Creates a hash table on the join key of the smaller table. Scans the larger table and looks up matching rows in the hash table. Hash join is very fast. The hash table is held in memory, so a hash join can use a lot of memory, depending on the size of the smaller table.

Sort merge join

The tables to be joined are sorted on the join attribute and then scanned in parallel to find the matching values. This is a good join method for tables that are too large to use a hash join.

Product join

Joins every qualifying row in the first table with every qualifying row in the second table. This type of join can be very expensive if spill files must be used.

Sort and Aggregate Operators

Sort

Sorts rows to prepare for operations such as an aggregation or merge join.

Group by

Groups rows by one or more columns.

Group / hash aggregate

Aggregates rows using a hash.

Motion Operators

Broadcast motion

Every segment sends its own local data to all other segment instances so that every segment instance has a complete local copy of the table.

Redistribution motion

Sends data from one table to another segment so that matching rows are located together, enabling a local join.

Gather motion

All segments send rows to the master where they are gathered into a single result set.

DML Operators

Assert

Performs constraints checking.

Split

Used for update operations.

Spill Files

Greenplum Command Center reports the total size for all spill files created for a query.

Greenplum Database creates spill files, also called workfiles, to save intermediate results when there is insufficient memory to execute a query in memory. Disk I/O is much slower than memory access, so a query that creates spill files will take longer to complete.

Investigating Spill File Usage

The `gp_toolkit` schema contains views you can use to see details about spill file usage for current queries. You can see the number and sizes of spill files created for each operator in a query execution plan, and totals by query and segment. This is useful information to detect data skew and to help tune queries.

See the [gp_toolkit Administrative Schema](#) reference for descriptions of these views.

Eliminating or Reducing Spill Files

You can work to eliminate spill files by increasing the amount of memory available to the query or by optimizing the query to use the memory available more efficiently.

You may be able to revise the query to prevent spilling by eliminating or postponing memory-intensive operators.

Following are some ways to increase memory available to queries when resource group resource management is enabled in Greenplum Database.

- Decrease the resource group's concurrency so that each query's share of memory increases.
- Increase the resource group's `MEMORY_SHARED_QUOTA` parameter to increase the amount of resource group shared memory.
- Decrease the percentage of memory allocated to all resource groups to increase the amount of global shared memory.

When resource queue resource management is active, Greenplum Database can detect and terminate “runaway” queries that consume a high percentage of available memory. You can prevent runaway queries by limiting the number of spill files created or the total size of spill files created. See the `gp_workfile_limit*` configuration parameters below for more information.

If you cannot prevent queries from spilling, it is important to ensure that the number of spill files created is minimized and that problems such as CPU or data skew are found and corrected. Skew can create excessive numbers of spill files on one or more segments.

To minimize disk usage and I/O when spill files are created, make sure the `gp_workfile_compression` configuration parameter is set to `on`. When enabled, Greenplum Database uses Zstandard compression for spill files.

Limiting Spill Files with Server Configuration Parameters

Greenplum Database by default limits the number of spill files allowed per query for each segment to 100,000. You can raise or lower this limit, and you can also limit the number of spill files for all queries on a segment, and limit the disk space consumed by spill files per query and per segment. Use the following Greenplum Database server configuration parameters to manage spill files.

`gp_workfile_limit_files_per_query`

Sets the maximum number of spill files allowed per query per segment. Default is 100,000.

`gp_workfile_limit_per_query`

Sets the maximum disk size an individual query is allowed to use for spill files at each segment. The default value is 0, which means no limit is enforced.

`gp_workfile_limit_per_segment`

Sets the maximum total disk size that all running queries are allowed to use for creating spill files at each segment. The default value is 0, which means a limit is not enforced.

`gp_workfile_compress_algorithm`

Specifies the compression algorithm to use for spill files when a hash aggregation or hash join operation spills to disk during query processing. The default is 'none'. Set to 'zlib' to enable compression. Using compression reduces the number of I/O operations at the expense of increased CPU.

See also [Managing Spill Files Generated by Queries](#).

Alert Help Topics

- Database connectivity failure
- Segment failure
- Average memory (segment hosts) exceeds [%] for [min]
- Memory (master) exceeds [%] for [N] minutes
- Total disk space exceeds [%] full
- Query is blocked for [N] minutes
- Number of connections exceeds [N]
- Average CPU (master) exceeds [%] for [N] min
- Out of memory errors
- Query runtime exceeds [N] minutes
- Average CPU (segment hosts) exceeds [%] for [N] minutes
- Spill files for a query exceeds [GB]

Database connectivity failure

What is this alert?

Command Center raises an alert if it is unable to connect to the Greenplum Database system. If Command Center is running on the Greenplum Database master host, it is likely that the database is down or not accepting connections.

If Command Center is running on the standby master host, the problem could be that the master host is down, there is a networking issue, or that the database is down or not accepting connections.

Command Center tries to connect to the database three times before issuing an alert.

What to do?

If Command Center is running on the Greenplum Database master host:

- Run the `gpstate` command-line utility on the Greenplum master to check the status of the database system. Correct any problems identified in the command output.

If Command Center is running on the Greenplum Database standby master host:

- Log in to the standby master host as the `gpadmin` user and then SSH to the master host.

```
$ ssh gpadmin@<master-hostname>
```


Be sure that <master-hostname> matches the `hostname` column in the `gp_segment_configuration` system table. If you are unable to reach the master host, make sure the host is running and resolve any networking issues.

- Run the `gpstate` command-line utility on the Greenplum master to check the status of the database system. Correct any problems identified in the command output.

Segment failure

What is this alert?

Command Center checks the status of the Greenplum Database segments every 30 seconds and raises an alert if any segments are down or running in their non-preferred roles. This alert will be raised hourly until an administrator has recovered the failed segments.

With segment mirroring enabled, Greenplum Database can tolerate a primary or mirror segment failure as long as there is a working instance for every segment in the cluster. If both the primary and mirror instances fail for any single segment, Greenplum Database cannot run queries. For this reason, it is important to recover the failed segment instance to protect from loss of service.

Segment instances have a “preferred role,” either primary or mirror. When all segment instances are in their preferred roles, each segment host has the same number of primary and mirror segments. If a primary instance fails, its mirror instance assumes the primary role and the distribution of primary segments is no longer balanced. This can slow down query processing because a host with more primary segments than others can take longer to complete queries.

What to do

Restore the failed segments and return the segments to their preferred roles so that the cluster is in balance.

See [Recovering From Segment Failures](#) for steps to recover Greenplum Database segments.

Average memory (segment hosts) exceeds [%] for [min]

What is this alert?

This alert warns of high memory consumption on the Greenplum Database segment hosts for an extended period.

Every 15 seconds, the Greenplum Database metrics collector extension samples the memory in use on each segment host. Memory used for kernel cache and buffers is excluded. The average for all segments is calculated. If the average percentage remains above the threshold that is set for the alert for the number of minutes specified, an alert is issued.

If memory consumption is increasing over time, queries could start to fail with out of memory errors.

What to do?

Check the **Query Monitor** to see if there is unusually heavy query activity.

Look for active queries that perform hash joins or sorts on a large number of tuples. If possible, optimize the queries to eliminate rows earlier so that these memory-intensive operations process a fewer number of tuples.

Adjust resource queues or resource groups to limit the number of concurrent transactions.

Use the `ps` command to identify non-database processes consuming excessive memory. Kill unnecessary processes or move them to another server.

Memory (master) exceeds [%] for [min]

What is this alert?

When the master host memory usage is higher than the specified percentage for more than the specified number of minutes, Command Center raises an alert.

Command Center samples the memory in use on the master host every fifteen seconds. Memory used for kernel buffers and cache is excluded from the calculation. An alert is raised if the samples collected during the number of minutes specified are all higher than the specified percentage.

What to do

Use the `ps` command to identify non-database processes consuming excessive memory and stop them or relocate them to other servers.

If Command Center is running on the master host, restart it on the standby master host.

Total disk space exceeds [%]

What is this alert?

This alert is raised when the percentage of segment host disk space in use exceeds the percentage specified in the alert rule. The master disk space is not included in the calculation. The alert is raised once a day until the percentage drops below the percentage in the alert rule.

What to do

This alert warns you so that you can add disk storage or free up storage in order to prevent a catastrophic disk full error that could interrupt Greenplum Database service.

Here are some suggestions for freeing space on Greenplum Database hosts.

- Archive and remove backup files
- Archive and drop older partitions
- Rotate OS and database log files
- Drop unneeded external tables and their data files
- Vacuum database tables and catalog tables

Query is blocked for [min]

What is this alert?

If a query that has started to execute is blocked by another query for the specified number of minutes, Command Center raises an alert.

Queries that are queued, but have not yet started, do not trigger this alert.

Alert details

The alert contains the Query ID, database name and user, and run-time details.

Qid":

```
{“Tmid”:1541113373,“Ssid”:6968,“Ccnt”:3},“Database”:“postgres”,“User”:“gpmon”,“SubmitTime”:“2018-11-02T16:10:04+08:00”,“StartTime”:“2018-11-02T16:10:04+08:00”,“QueryText”:“”}
```

Need alert type specific JSON example to write this out

What to do

1. Use the **Query Monitor** to locate the blocked query and the query that blocks it.
2. Determine whether the blocking query is executing properly:

```
- Is the query also blocked?
- Is the query blocking a large number of other queries?
- Is the query creating excessive spill files?
- Is the query running in the correct resource group or resource queue?
- Is the query running longer than usual?
- Does the query have excessive data or CPU skew?
```

1. Determine whether you should allow the query to complete, or cancel the query so that the blocked queries can resume.

Number of connections exceeds [n]

What is this alert?

This alert is raised when the number of concurrent connections at the Greenplum Database master instance exceeds a specified number. The number specified should be set lower than the `max_connections` server configuration parameter so that when you receive the alert you can act before Greenplum Database begins to reject client connection requests. For example, if `max_connections` for the master is set to 100, you could set an alert to 80.

What to do

Look for idle or stale connections and terminate them

Users can connect to Greenplum Database using a client such as `psql`, execute queries and remain

connected, but inactive, leaving the connection in an idle state. Greenplum Database eventually releases resources used by idle connections, but once the maximum number of allowed connections has been reached, new connection requests are denied.

Use the `pg_stat_activity` system view to find idle connections.

```
SELECT datname, procpid, sess_id, username, current_query from pg_stat_activity;
```

Use the `pg_cancel_backend(<PID>)` function to cancel idle connections.

Determining *how long* a connection has been idle is not possible with just the information in the Greenplum Database 5.x system tables. You can, however, see this information by creating the `session_level_memory_consumption` view in the database. Follow the instructions at [Viewing Session Memory Usage Information](#) to create this view in each database.

After you install the `session_level_memory_consumption` view, a query like the following shows the idle connections with the length of time they have been idle.

```
SELECT a.datname, application_name, a.username, a.sess_id, procpid,
       now()-idle_start AS time_idle
FROM pg_stat_activity a, session_state.session_level_memory_consumption b
WHERE a.sess_id = b.sess_id AND b.idle_start < now();
ORDER BY time_idle DESC;
```

CPU (master) exceeds [%] for [min]

What is this alert?

The metrics collector extension checks CPU utilization on the master host every 15 seconds. If the percentage of CPU in use is higher than the percentage specified in the alert rule for the number of minutes specified in the rule, Command Center raises an alert.

The Greenplum Database master uses the greatest amount of CPU at the start of a query, while planning the query, and at the end of the query, while gathering results from the segments. For a large result set it is normal to see a spike in the query's CPU usage on the master during the gather operation. With many queries running concurrently, the CPU spikes should even out.

What to do

Begin by viewing the Command Center **Query Monitor** and **Query Details** views to see if there are one or more very large queries nearing completion, or if the high usage can be explained by unusual workloads or heavy query activity.

If the query monitor offers no explanation for high CPU usage, you should investigate master host processes using Linux utilities such as `ps` and `top` to identify processes consuming the CPU. If the process ID of an errant process is a `postgres` process, you can query the `pg_stat_activity` system table to find the query and, if needed, use the `pg_terminate_backend()` function to terminate the query.

See also

Investigating a Performance Problem

Out of memory error

What is this alert?

If a query requests additional memory and is denied, the query fails with an out of memory error and an alert is raised.

What to do

Greenplum Database has two ways to manage memory resources: resource queues and resource groups. Resource queues deal primarily with fixed quantities of memory, where resource groups deal with portions—percentages—of available memory.

If you use resource groups to manage memory, you can use the **Admin> Workload Mgmt** view to adjust them so that more memory is available to queries that are failing due to out of memory errors. If you use resource queues, you use the `CREATE RESOURCE QUEUE` and `ALTER RESOURCE QUEUE` SQL commands to configure them.

There are many factors to consider when allocating memory for queries, including configuring the operating system, allocating a share of memory to Greenplum Database, and configuring a set of resource queues or resource groups to share the memory available to Greenplum Database.

For complete information about how Greenplum Database manages memory and how to configure it, see:

- [Using Resource Groups](#)
- [Using Resource Queues](#)

Query runtime exceeds [min]

What is this alert?

An alert is raised if the total runtime for a query is greater than the specified number of minutes. The alert is raised once per query.

Run time is calculated from the time the query begins to execute. The time the query was queued is excluded.

Alert details

- query ID
- database name
- user name
- time the query was submitted
- time the query started

What to do

Use the **Query Monitor** to check the execution status of the query.

If the query is blocked, investigate the queries that hold the locks.

Average CPU (segment hosts) exceeds [%] for [min]

What is this alert?

Command Center samples CPU usage on all segment hosts every 15 seconds and calculates the average CPU usage for the cluster. An alert is raised if the average CPU usage is greater than the specified percentage for longer than the specified number of minutes.

What to do

Use the Command Center **Query Monitor** to identify currently running individual queries with high CPU usage.

Use the Command Center **History** view to see CPU load during the period prior to the alert and identify completed queries using too much CPU.

Check CPU usage using operating system utilities such as `ps` and `top` to identify any operating system processes that are consuming excessive CPU, for example backup, restore, or ETL processes.

Spill files for a query exceeds [GB]

What is this alert?

Command Center raises an alert if the combined size of spill files for any query exceeds the specified number of gigabytes. This alert is raised just once per query.

Greenplum Database creates spill files to temporarily store data on disk when the data exceeds the amount of memory allocated to the operation. Because memory I/O is much faster than disk I/O, a query that creates spill files takes longer to complete than it would if there was sufficient memory available to avoid creating spill files.

What to do

Use the Command Center **Query Monitor** to view the plan for the query identified in the alert.

If possible, revise the query so that more rows are eliminated earlier in the plan, eliminating or reducing the size of spill files.

Consider reconfiguring the resource queue or resource group that manages the query to make more memory available to the query. If you use resource groups to manage resources, you can use the Command Center **Admin> Workload Mgmt** view to modify resource allocations.

For more information

- [Managing Spill Files Generated by Queries](#)
- [Using Resource Queues](#) for information about configuring Greenplum Database resource queues
- [Using Resource Groups](#) for information about configuring resource groups