Changelog

Contents

pgMonitor

{{< logo src="/images/pgmonitor_logo.svg" text="Crunchy Monitoring" >}}

pgMonitor is your all-in-one tool to easily create an environment to visualize the health and performance of your PostgreSQL cluster.

pgMonitor combines a suite of tools to facilitate the collection and visualization of important metrics that you need be aware of in your PostgreSQL database and your host environment, including:

- Connection counts: how busy is your system being accessed and if connections are hanging
- Database size: how much disk your cluster is using
- Replication lag: know if your replicas are falling behind in loading data from your primary
- Transaction wraparound: don't let your PostgreSQL database stop working
- Bloat: how much extra space are your tables and indexes using
- System metrics: CPU, Memory, I/O, uptime

pgMonitor is also highly configurable, and advanced users can design their own metrics, visualizations, and add in other features such as alerting.

Running pgMonitor will give you confidence in understanding how well your PostgreSQL cluster is performing, and will provide you the information to make calculated adjustments to your environment.

Contents

Purpose

pgMonitor is an open-source monitoring solution for PostgreSQL and the systems that it runs on. pgMonitor came from the need to provide a way to easily create a visual environment to monitor all the metrics a database administrator needs to proactively ensure the health of the system.

pgMonitor combines multiple open-source software services to create a robust PostgreSQL monitoring environment. These include:

- Prometheus - an open-source metrics collector that is highly customizable.
- Grafana - an open-source data visualizer that allows you to generate many different kinds of charts and graphs.
- SQL Exporter - an open-source exporter for Prometheus that supports collecting metrics from multiple database systems including PostgreSQL.
- pgMonitor extension - a PostgreSQL extension that provides a means to collect metrics that can be used by an external collection source.

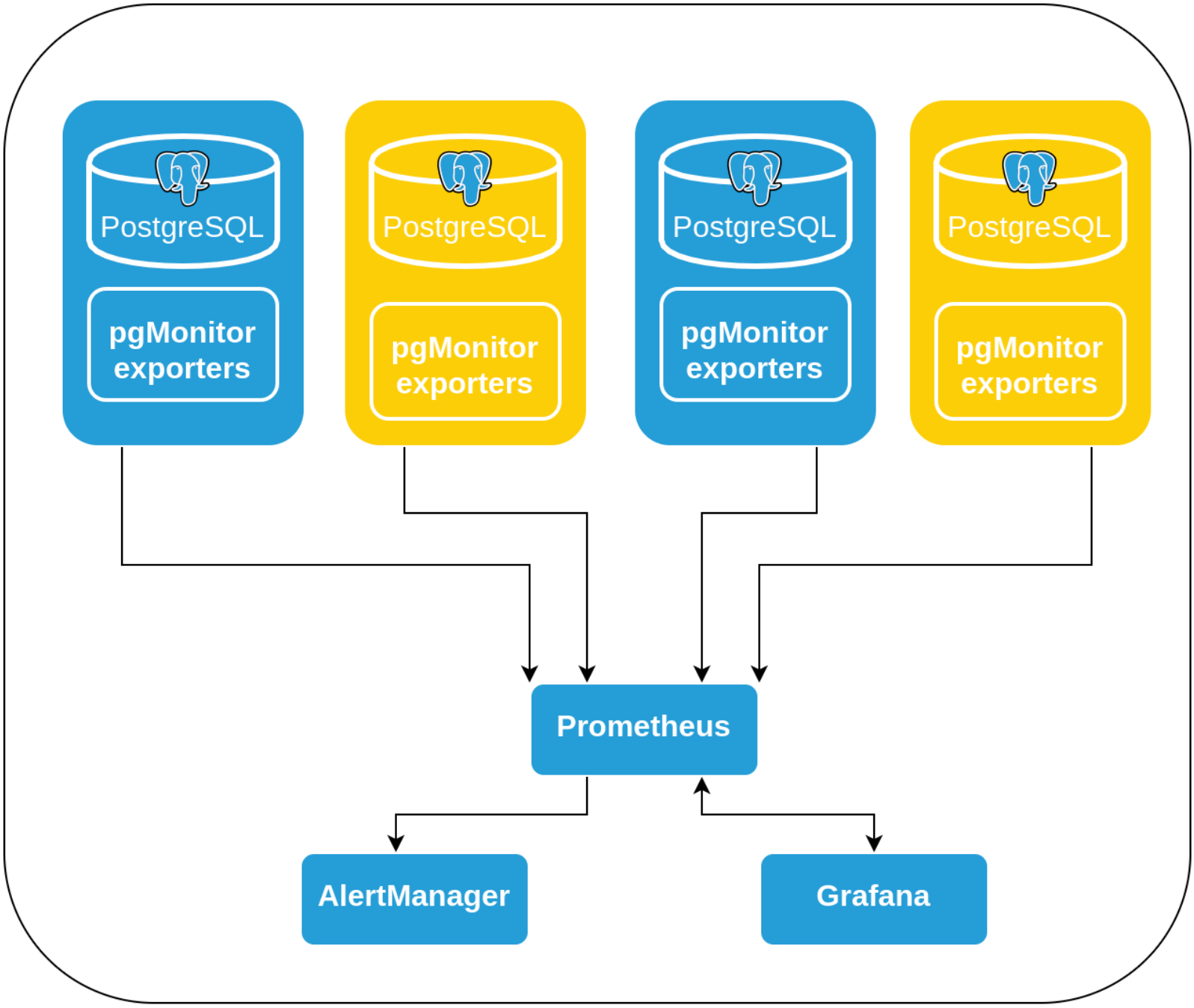

Figure 1: pgMonitor

Supported Platforms

pgMonitor Extension

- As of pgMonitor version 5.2, the pgMonitor extension version requirement is a minimum of 2.1.0

Operating Systems

- RHEL 8/9 (Build/Run Testing, Setup Instructions)
- Ubuntu 20/22 (Build/Run Testing)

PostgreSQL

- pgMonitor plans to support all PostgreSQL versions that are actively supported by the PostgreSQL community. Once a major version of PostgreSQL reaches its end-of-life (EOL), pgMonitor will cease supporting that major version soon after. Please see the official PostgreSQL website for community supported releases.

etcd

- Version 3.5 and greater is supported for the Grafana dashboard

HAProxy

- Version 2.x for Grafana dashboard

PgBouncer

- PgBouncer 1.21+
- pgbouncer_fdw 1.1.0 (optional with sql_exporter)

Installation

Installation instructions for each package are provided in that packages subfolder. Each step in the installation process is listed here, with a link to additional to further installation instructions for each package.

1. **exporter**
2. **Prometheus**
3. **Grafana**

Notes on upgrading can be found in each relevant section.

Version History

For the full history of pgMonitor, please see the CHANGELOG.

Sponsors

{{< logo src="/images/crunchy_logo.png" text="Crunchy Data" >}}

Crunchy Data is pleased to sponsor pgMonitor and many other open-source projects to help promote support the PostgreSQL community and software ecosystem.

Legal Notices

The Linux instructions below use RHEL, but any Linux-based system should work. Crunchy Data customers can obtain Linux packages through the Crunchy Customer Portal.

IMPORTANT NOTE: As of pgMonitor version 5.0.0, postgres_exporter has been deprecated in favor of sql_exporter. Support for postgres_exporter is still possible with 5.0, but only for bug fixes while custom queries are still supported upstream. No new features will be added using postgres_exporter and it will be fully obsoleted in a future version of pgMonitor. We recommend migrating to sql_exporter as soon as possible.

Installation

Package Install

The following RPM packages are available to Crunchy Data customers through the Crunchy Customer Portal. To access the pgMonitor packages, please follow the same instructions for setting up access to the Crunchy Postgres packages.

After installing via these packages, continue reading at the Setup section.

Available Packages

Package Name	Description
blackbox-exporter	Package for the blackbox_exporter
node-exporter	Base package for node_exporter
pg-bloat-check	Package for pg_bloat_check script
pgmonitor-blackbox-exporter-extras	Crunchy-optimized configurations for blackbox_exporter
pgmonitor-node_exporter-extras	Crunchy-optimized configurations for node_exporter
pgmonitor-pg##-extension	Crunchy monitoring PostgreSQL extension used by sql_exporter
pgmonitor-sql-exporter-extras	Crunchy-optimized configurations for sql_exporter
sql-exporter	Base package for sql_exporter

Non-Package Install

For non-package installations on Linux, applications can be downloaded from their respective repositories:

Application	Source Repository
blackbox_exporter	https://github.com/prometheus/blackbox_exporter
node_exporter	https://github.com/prometheus/node_exporter
pg_bloat_check	https://github.com/keithf4/pg_bloat_check
pgmonitor-extension	https://github.com/CrunchyData/pgmonitor-extension
sql_exporter	https://github.com/burningalchemist/sql_exporter

User and Configuration Directory Installation You will need to create a user named {{< shell >}}ccp_monitoring{{< /shell >}} which you can do with the following command:

```
sudo useradd -m -d /var/lib/ccp_monitoring ccp_monitoring
```

Configuration File Installation All executables installed via the above releases are expected to be in the {{< shell >}}/usr/bin{{< /shell >}} directory. A base node_exporter systemd file is expected to be in place already. An example one can be found here:

https://github.com/prometheus/node_exporter/tree/master/examples/systemd

A base blackbox_exporter systemd file is also expected to be in place. No examples are currently available.

The files contained in this repository are assumed to be installed in the following locations with the following names. In the instructions below, you should replace a double-hash (##) with the two-digit major version of PostgreSQL you are running (ex: 12, 13, 14, etc.).

node_exporter The {{< shell >}}node_exporter{{< /shell >}} data directory should be {{< shell >}}/var/lib/ccp_monitoring/node_e
/shell >}} and owned by the {{< shell >}}ccp_monitoring{{< /shell >}} user. You can set it up with:

```
sudo install -m 0700 -o ccp_monitoring -g ccp_monitoring -d /var/lib/ccp_monitoring/node_exporter
```

The following pgMonitor configuration files should be placed according to the following mapping:

pgmonitor Configuration File	System Location
node_exporter/linux/crunchy-node-exporter-service-rhel.conf	/etc/systemd/system/node_exporter.service.d/cru node-exporter-service-rhel.conf
node_exporter/linux/sysconfig.node_exporter	/etc/sysconfig/node_exporter

sql_exporter sql_exporter takes advantage of the Crunchy Data pgmonitor-extension (https://github.com/CrunchyData/pgmonitor-extension) to provide a much easier configuration and setup. The extension takes care of creating all the necessary objects inside the database.

The minimum required version of pgmonitor-extension is currently 2.1.0.

The following pgMonitor configuration files should be placed according to the following mapping:

pgMonitor Configuration File	System Location
sql_exporter/common/*.yml	/etc/sql_exporter/*.yml
sql_exporter/common/*.sql	/etc/sql_exporter/*.sql
sql_exporter/linux/crunchy-sql-exporter@.service	/usr/lib/systemd/system/crunchy-sql-exporter@.service
sql_exporter/linux/sql_exporter.sysconfig	/etc/sysconfig/sql_exporter
sql_exporter/linux/crontab.txt	/etc/sysconfig/crontab.txt
postgres_exporter/linux/pgbackrest-info.sh	/usr/bin/pgbackrest-info.sh
postgres_exporter/linux/pgmonitor.conf	/etc/pgmonitor.conf (multi-backrest-repository/container environment only)
sql_exporter/common/sql_exporter.yml.example	/etc/sql_exporter/sql_exporter.yml

blackbox_exporter Note that blackbox_exporter is typically installed on the Prometheus node and does not need to be installed on any other nodes in the default pgMonitor configuration. The following pgMonitor configuration files should be placed according to the following mapping:

pgMonitor Configuration File	System Location
blackbox_exporter/common/blackbox_exporter.sysconfig	/etc/sysconfig/blackbox_exporter
blackbox_exporter/common/crunchy-blackbox.yml	/etc/blackbox_exporter/crunchy-blackbox.yml

Upgrading

- If you are upgrading to version 5.0 and transitioning to using the new sql_exporter, please see the documentation in Upgrading to pgMonitor v5.0.0
- See the CHANGELOG for full details on both major & minor version upgrades.
- It is recommended to re-run the `setup_db.sql` script after running any updates to the `pgmonitor-extension` to ensure any new needed privileges are granted and that new objects have their permissions set properly. If you have any other roles besides the default interacting with the `pgmonitor-extension` objects, you may have to adjust their permissions as well.

Setup

Setup on RHEL

Service Configuration The following files contain defaults that should enable the exporters to run effectively on your system for the purposes of using pgMonitor. Please take some time to review them.

If you need to modify them, see the notes in the files for more details and recommendations: - {{< shell >}}/etc/systemd/system/node_expor node-exporter-service-rhel.conf{{< /shell >}} - {{< shell >}}/etc/sysconfig/node_exporter{{< /shell >}} - {{< shell >}}/etc/sysconfig/sql_exporter{{< /shell >}}

Database Configuration

General Configuration First, make sure you have installed the PostgreSQL contrib modules. An example of installing this on a RHEL system would be:

```
sudo yum install postgresql##-contrib
```

Where `##` corresponds to your current PostgreSQL version.

You will need to modify your {{< shell >}}postgresql.conf{{< /shell >}} configuration file to tell PostgreSQL to load the following shared libraries:

```
shared_preload_libraries = 'pg_stat_statements,auto_explain,pgmonitor_bgw'
```

You will need to restart your PostgreSQL instance for the change to take effect. pgMonitor has optional metrics that can be collected via pg_stat_statements. auto_explain does not do anything to your database without further configuration. But even if neither of these extensions are initially used, they are very good to have enabled here by default for when they may be needed in the future.

The pgmonitor-extension uses its own background worker to refresh certain metric data. It also requires that you set which databases will have metrics monitoring enabled. You do this with the `pgmonitor_bgw.dbname` GUC in your postgresql.conf. At a minimum the "global" database needs to be mentioned, typically `postgres`, but if per-database metrics for other databases are also desired (tables statistics, bloat, etc), they must be listed here in CSV format. This value can be changed at any time with just a reload of PostgreSQL.

```
pgmonitor_bgw.dbname = 'postgres,alphadb,betadb,etc'
```

The following statement only needs to be run on the "global" database. If you want the pg_stat_statements view to be visible in other databases, this statement must be run there as well.

```
CREATE EXTENSION pg_stat_statements;
```

Monitoring Setup

Configuration File	Description
setup_db.sql	Creates `ccp_monitoring` role with all necessary grants. Creates the pgmonitor-extension and sets proper privileges
sql_exporter.yml.example	Example configuration file for configuring sql_exporter to connect to PostgreSQL and setting collection files to use
crunchy_backrest_collector.yml	Collection file with pgBackRest queries and metrics
crunchy_bloat_check_collector.yml	Collection file with pg_bloat_check queries and metrics
crunchy_global_collector.yml	Collection file with global level queries and metrics
crunchy_per_db_collector.yml	Collection file with general per-database level queries and metrics
crunchy_pg_stat_statements_collector.yml	Collection file with pg_stat_statements queries and metrics
crunchy_pg_stat_statements_reset_collector.yml	Collection file with options to allow resetting of pg_stat_statements metrics
crunchy_pgbouncer_collector_*.yml	Collection files for pgBouncer queries and metrics. The number suffix on the filename is the minimum version of pgBouncer required for that file. If there is a newer version file available, that one must be used if that version of pgBouncer or greater is running. Example: collector_121.yml requires at least pgBouncer 1.21. But there is also a collector_124.yml file, so if you are running pgBouncer 1.24+, you must use the 124 file. Note that a newer file will only be created if necessary to support changes in the pgBouncer API.

Run the `setup_db.sql` file on all databases that will be monitored by pgMonitor. At minimum this must be at least the global database so the necessary database objects are created. The `pgmonitor-extension` is expected to be available to be installed in the target database(s) when running this file. Note the `setup_db.sql` file is a convenience file and the steps contained within it can be done manually and customized as needed. Note that a default password is not set for the `ccp_monitoring` database role.

The `sql_exporter.yml.example` file should be copied and renamed to `sql_exporter.yml` since this is what the sysconfig file is expecting to find. This file contains settings for sql_exporter, the list of collection files to use, and the configuration for which databases to connect to and which collections to run on each database. Please see the examples inside the file and refer to the upstream project for all of the configuration options available. The example shows how to run both the global and per-db collections on the default 'postgres' database. It also shows how you can connect to PgBouncer to collect metrics directly from it as well. The collector names that can be used can be found inside the collection files at the top. For additional information on setting up the sql_exporter, please see the upstream documentation.

Note that your `pg_hba.conf` will have to be configured to allow the {{< shell >}}ccp_monitoring{{< /shell >}} system user to connect as the {{< shell >}}ccp_monitoring{{< /shell >}} role to any database in the instance. sql_exporter is set to connect via the local TCP loopback by default. If passwordless login is desired, a .pgpass file can be created for the ccp_monitoring user or the connection configuration can be changed to use a local socket and peer-based authentication can be done instead.

For replica servers, the setup is the same except that the setup_db.sql file does not need to be run since writes cannot be done there and it was already run on the primary.

Enabling/Disabling Extension Metrics As mentioned above, metrics that sql_exporter actively tries to collect are maintained by the collection files that are added to the configuration. However the pgmonitor-extension also has some settings to enable/disable the refresh of the materialized views that it maintains. Not all are enabled by default, so it is good to review the `active` column in both the `metric_matviews` and `metric_tables` configuration tables. For example, in `metric_tables`, the pgBackRest data is not enabled by default since not everyone will have that backup solution in place.

```
postgres=# SELECT * FROM pgmonitor_ext.metric_tables ;
-[ RECORD 1 ]-----+---------------------------------------
table_schema      | pgmonitor_ext
table_name        | pgbackrest_info
refresh_statement | SELECT pgmonitor_ext.pgbackrest_info()
run_interval      | 00:10:00
last_run          |
last_run_time     |
```

```
active            | f
scope             | global
```

In order to enable that one, simply set the `active` column to true and during the next refresh cycle, it will attempt to refresh the table that stores those metrics. You'll know it's successfully running if you see the `last_run` columns being set. If not, please check logs for any errors.

```
postgres=# UPDATE pgmonitor_ext.metric_tables SET active = 'true';
UPDATE 1

postgres=# SELECT * FROM pgmonitor_ext.metric_tables ;
-[ RECORD 1 ]-----+---------------------------------------
table_schema      | pgmonitor_ext
table_name        | pgbackrest_info
refresh_statement | SELECT pgmonitor_ext.pgbackrest_info()
run_interval      | 00:10:00
last_run          | 2024-08-16 09:48:27.328069-04
last_run_time     | 00:00:00.016229
active            | t
scope             | global
```

Further details on maintaining the pgmonitor-extension can be found in its documentation (linked above)

Access Control: GRANT statements The {{< shell >}}ccp_monitoring{{< /shell >}} database role (created by running the `setup_db.sql` file above) must be allowed to connect to all databases in the cluster. Note that by default, all users are granted CONNECT on all new databases, so this step can likely be skipped. Otherwise, run the following command to generate the necessary GRANT statements:

```
SELECT 'GRANT CONNECT ON DATABASE "' || datname || '" TO ccp_monitoring;'
FROM pg_database
WHERE datallowconn = true;
```

This should generate one or more statements similar to the following:

```
GRANT CONNECT ON DATABASE "postgres" TO ccp_monitoring;
```

Run these grant statements to then allow monitoring to connect.

Bloat setup Run the script on the specific database(s) you will be monitoring for bloat in the cluster. See the note below, or in crontab.txt, concerning special privilege requirements for using this script.

```
psql -d postgres -c "CREATE EXTENSION pgstattuple;"
/usr/bin/pg_bloat_check.py -c "host=localhost dbname=postgres user=postgres" --create_stats_table
psql -d postgres -c "GRANT SELECT,INSERT,UPDATE,DELETE,TRUNCATE ON bloat_indexes, bloat_stats,
   bloat_tables TO ccp_monitoring;"
```

The {{< shell >}}/etc/sql_exporter/##/crontab.txt{{< /shell >}} file has an example bloat check crontab entry. Modify this example to schedule bloat checking weekly during your 'off-peak' hours; alternatively, scheduling it monthly is usually good enough for most databases as long as the results are acted upon quickly.

{{< note >}}Bloat monitoring requires the user running the check to be able to read all possible tables that will ever exist. PostgreSQL 14 introduced the built-in role {{< shell >}}pg_read_all_data{{< /shell >}} that can be granted to any role to allow it to read all possible data for the entire cluster. It is recommended to grant this role vs running the bloat check as a superuser. If you are running a version of PostgreSQL less than 14, a superuser is required and you will have to adjust the crontab accordingly to run as that user.

```
GRANT pg_read_all_data TO ccp_monitoring;
```

{{< /note >}}

Blackbox Exporter The configuration file for the blackbox_exporter provided by pgMonitor ({{< shell >}}/etc/blackbox_exporter/crun blackbox.yml{{< /shell >}}) provides a probe for monitoring any IPv4 TCP port status. The actual target and port being monitored are controlled via the Prometheus target configuration system. Please see the pgMonitor Prometheus documentation for further details. If any additional Blackbox probes are desired, please see the upstream documentation.

PGBouncer It is possible for sql_exporter to connect directly to pgBouncer to collect metrics. Specific settings must be used and the example sql_exporter configuration and relevant collection file(s) have these settings enabled. Please refer to those files.

```
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
sudo systemctl status node_exporter
```

```
sudo systemctl enable crunchy-sql-exporter@sql_exporter
sudo systemctl start crunchy-sql-exporter@sql_exporter
sudo systemctl status crunchy-sql-exporter@sql_exporter
```

To allow the possible use of multiple sql_exporters running on a single system, and to avoid maintaining many similar service files, a systemd template service file is used. Note that is is NOT necessary to run multiple sql_exporters to monitoring multiple databases in a single instance of PostgreSQL. Nor is it even necessary for multiple exporters when monitoring multiple PostgreSQL instances. A single sql_exporter can connect to as many target PostgreSQL instances as you need, so in general multiple will not be needed.

The name of the sysconfig EnvironmentFile to be used by the service is passed as the value after the "@" and before ".service" in the service name. The default exporter's sysconfig file is named "sql_exporter". If you need to run multiple sql_exporters on a single system, simply make a new copy of the sysconfig file and pass that to the service name.

```
sudo systemctl enable crunchy-sql-exporter@sql_exporter_cluster2
sudo systemctl start crunchy-sql-exporter@sql_exporter_cluster2
sudo systemctl status crunchy-sql-exporter@sql_exporter_cluster2
```

If you've installed the blackbox exporter on the Prometheus node:

```
sudo systemctl enable blackbox_exporter
sudo systemctl start blackbox_exporter
sudo systemctl status blackbox_exporter
```

Monitoring multiple databases

sql_exporter can connect to as many databases as you need. Simply add another connection configuration to the `job_name` in the sql_exporter configuration file for the other databases you wish to monitor. If making use of pgMonitor's metrics, ensure that the pgmonitor-extension is also installed on those target databases.

IMPORTANT NOTE: If you are collecting metrics on multiple databases with the same exporter, you must ensure the results of the queries for those metrics are unique for each row. For example, you should add the database name to all queries and set that as a label to ensure the metrics are unique for each database. Otherwise you will get duplicate metric errors or confusing results. All per-database metrics provided by pgMonitor have been set to ensure unique values.

```
jobs:
  - job_name: global_targets
    collectors: [crunchy_global]
    static_configs:
        - targets:
            pg1: 'pg://ccp_monitoring@127.0.0.1:5432/postgres?sslmode=disable'
  - job_name: per_db_targets
    collectors: [crunchy_per_db]
    static_configs:
        - targets:
            postgres: 'pg://ccp_monitoring@127.0.0.1:5432/postgres?sslmode=disable'
            alpha: 'pg://ccp_monitoring@127.0.0.1:5432/alpha?sslmode=disable'
            beta: 'pg://ccp_monitoring@127.0.0.1:5432/beta?sslmode=disable'
            charlie: 'pg://ccp_monitoring@127.0.0.1:5432/charlie?sslmode=disable'
```

Metrics Collected

The metrics collected by our exporters are outlined below.

PostgreSQL

PostgreSQL metrics are collected by sql_exporter. pgMonitor uses custom queries for its PG metrics.

Common Metrics These metrics are common to all versions of PostgreSQL and are recommended as a minimum default for the global exporter exporter connection.

- *ccp_archive_command_status_seconds_since_last_fail* - Seconds since the last `archive_command` run failed. If zero, the `archive_command` is succeeding without error.
- *ccp_database_size_bytes* - Total size of each database in PostgreSQL instance
- *ccp_is_in_recovery_status* - Current value of the pg_is_in_recovery() function expressed as 1 for true (instance is a replica) and 2 for false (instance is a primary)
- *ccp_connection_stats_active* - Count of active connections
- *ccp_connection_stats_idle* - Count of idle connections
- *ccp_connection_stats_idle_in_txn* - Count of idle in transaction connections
- *ccp_connection_stats_max_blocked_query_time* - Runtime of longest running query that has been blocked by a heavyweight lock
- *ccp_connection_stats_max_connections* - Current value of max_connections for reference
- *ccp_connection_stats_max_idle_in_txn_time* - Runtime of longest idle in transaction (IIT) session.
- *ccp_connection_stats_max_query_time* - Runtime of longest general query (inclusive of IIT).
- *ccp_connection_stats_max_blocked_query_time* - Runtime of the longest running query that has been blocked by a heavyweight lock
- *ccp_locks_count* - Count of active lock types per database
- *ccp_pg_hba_checksum_status* - Value of checksum monitioring status for pg_catalog.pg_hba_file_rules (pg_hba.conf). 0 = valid config. 1 = settings changed. Settings history is available for review in the table `monitor.pg_hba_checksum`. To reset current config to valid after alert, run monitor.pg_hba_checksum_set_valid(). Note this will clear the history table.
- *ccp_postmaster_uptime_seconds* - Time interval in seconds since PostgreSQL database was last restarted
- *ccp_postgresql_version_current* - Version of PostgreSQL that this exporter is monitoring. Value is the 6 digit integer returned by the `server_version_num` PostgreSQL configuration variable to allow easy monitoring for version changes.
- *ccp_replication_lag_replay_time* - Time since a replica received and replayed a WAL file; only shown on replica instances. Note that this is not the main way to determine if a replica is behind its primary. This metric only monitors the time since the replica replayed the WAL vs when it was received. It also does not monitor when a WAL replay replica completely stops receiving WAL (see received_time metric). It is a secondary metric for monitoring WAL replay on the replica itself. This metric always returns zero on a primary.
- *ccp_replication_lag_received_time* - Similar to *ccp_replication_lag_replay_time*, however this value always increases between replay of WAL files. Effective for monitoring that a WAL replay replica has actually received WAL files. Note this will cause false positives when used as an alert for replica lag if the primary receives little to no writes (which means there is no WAL to send). This metric always returns zero on a primary.
- *ccp_replication_lag_size_bytes* - Only provides values on instances that have attached replicas (primary, cascading replica). Tracks byte lag of every streaming replica connected to this database instance. This is the main way that replication lag is monitored. Note that if you have WAL replay only replicas, this will not be reflected here.
- *ccp_replication_slots_active* - Active state of given replication slot. 1 = true. 0 = false.
- *ccp_replication_slots_retained_bytes* - The amount of WAL (in bytes) being retained for given slot.
- *ccp_sequence_exhaustion_count* - Checks for any sequences that may be close to exhaustion (by default greater than 75% usage). Note this checks the sequences themselves, not the values contained in the columns that use said sequences. Function `monitor.sequence_status()` can provide more details if run directly on database instance.
- *ccp_settings_pending_restart_count* - Number of settings from pg_settings catalog in a pending_restart state. This value is from the similarly named column found in pg_catalog.pg_settings.
- *ccp_wal_activity_total_size_bytes* - Current size in bytes of the WAL directory
- *ccp_wal_activity_last_5_min_size_bytes* - Current size in bytes of the last 5 minutes of WAL generation. Includes recycled WALs.

The meaning of the following `ccp_transaction_wraparound` metrics, and how to manage when they are triggered, is covered more extensively in this blog post: https://info.crunchydata.com/blog/managing-transaction-id-wraparound-in-postgresql

- *ccp_transaction_wraparound_percent_towards_emergency_autovac* - Recommended thresholds set to 75%/95% when first evaluating vacuum settings on new systems. Once those have been reviewed and at least one instance-wide vacuum has been run, recommend thresholds of 110%/125%. Reaching 100% is not a cause for immediate concern, but alerting above 100% for extended periods of time means that autovacuum is not able to keep up with current transaction rate and needs further tuning.

- *ccp_transaction_wraparound_percent_towards_wraparound* - Recommend thresholds set to 50%/75%. If any of these thresholds is tripped, current vacuum settings must be evaluated and tuned ASAP. If critical threshold is reached, it is vitally important that vacuum be run on tables with old transaction IDs to avoid the cluster being forced to shut down for extended offline maintenance.

The following `ccp_stat_bgwriter` metrics are statistics collected from the pg_stat_bgwriter view for monitoring performance. These metrics cover important performance information about flushing data out to disk. Please see the documentation for further details on these metrics.

- *ccp_stat_bgwriter_buffers_alloc*
- *ccp_stat_bgwriter_buffers_backend*
- *ccp_stat_bgwriter_buffers_backend_fsync*
- *ccp_stat_bgwriter_buffers_checkpoint*
- *ccp_stat_bgwriter_buffers_clean*

The following `ccp_stat_database_*` metrics are statistics collected from the pg_stat_database view.

- *ccp_stat_database_blks_hit*
- *ccp_stat_database_blks_read*
- *ccp_stat_database_conflicts*
- *ccp_stat_database_deadlocks*
- *ccp_stat_database_tup_deleted*
- *ccp_stat_database_tup_fetched*
- *ccp_stat_database_tup_inserted*
- *ccp_stat_database_tup_returned*
- *ccp_stat_database_tup_updated*
- *ccp_stat_database_xact_commit*
- *ccp_stat_database_xact_rollback*

PostgreSQL Version Specific Metrics The following metrics either require special considerations when monitoring specific versions of PostgreSQL, or are only available for specific versions. Unless otherwise noted, the below metrics are available for all versions of PG. These metrics are recommend as a minimum default for the global exporter.

- *ccp_data_checksum_failure_count* - PostgreSQL 12 and later only. Total number of checksum failures on this database.
- *ccp_data_checksum_failure_time_since_last_failure_seconds* - PostgreSQL 12 and later only. Time interval in seconds since the last checksum failure was encountered.

Backup Metrics Backup monitoring only covers pgBackRest at this time. These metrics only need to be collected once per PostgreSQL instance so should be collected by the global exporter connection.

- *ccp_backrest_last_full_backup_time_since_completion_seconds* - Time since completion of last pgBackRest FULL backup
- *ccp_backrest_last_diff_backup_time_since_completion_seconds* - Time since completion of last pgBackRest DIFFERENTIAL backup. Note that FULL backup counts as a successful DIFFERENTIAL for the given stanza.
- *ccp_backrest_last_incr_backup_time_since_completion_seconds* - Time since completion of last pgBackRest INCREMENTAL backup. Note that both FULL and DIFFERENTIAL backups count as a successful INCREMENTAL for the given stanza.
- *ccp_backrest_last_info_runtime_backup_runtime_seconds* - Last successful runtime of each backup type (full/diff/incr).
- *ccp_backrest_last_info_repo_backup_size_bytes* - Actual size of only this individual backup in the pgbackrest repository
- *ccp_backrest_last_info_backup_error* - Count of errors tracked for this backup. Note this does not track incomplete backups, only errors encountered during the backup (checksum errors, file truncation, invalid headers, etc)

Per-Database Metrics These are metrics that are only available on a per-database level. These metrics are optional and recommended for the non-global, per-db exporter connection. They can be included in the global exporter as well if the global database needs per-database metrics monitored. Please note that depending on the number of objects in your database, collecting these metrics can greatly increase the storage requirements for Prometheus since all of these metrics are being collected for each individual object.

- *ccp_table_size_size_bytes* - Table size inclusive of all indexes in that table

The following `ccp_stat_user_tables_*` metrics are statistics collected from the pg_stat_user_tables. Please see the PG documentation for descriptions of these metrics.

- *ccp_stat_user_tables_analyze_count*
- *ccp_stat_user_tables_autoanalyze_count*
- *ccp_stat_user_tables_autovacuum_count*
- *ccp_stat_user_tables_n_tup_del*
- *ccp_stat_user_tables_n_tup_ins*
- *ccp_stat_user_tables_n_tup_upd*
- *ccp_stat_user_tables_vacuum_count*

Bloat Metrics Bloat metrics are only available if the `pg_bloat_check` script has been setup to run. See instructions above. These metrics are per-database so, should be used by the per-db exporter connection.

- *ccp_bloat_check_size_bytes* - Size of object in bytes
- *ccp_bloat_check_total_wasted_space_bytes* - Total wasted space in bytes of given object

pgBouncer Metrics The following metric prefixes correspond to the SHOW command views found in the pgBouncer documentation. Each column found in the SHOW view is a separate metric under the respective prefix. Ex: `ccp_pgbouncer_pools_client_active` corresponds to the `SHOW POOLS` view's `client_active` column.

sql_exporter can connect directly to pgBouncer with some specific configuration options set. See the example `sql_exporter.yml` and the `crunchy_pgbouncer_collector_###.yml` file.

- *ccp_pgbouncer_pools* - SHOW POOLS
- *ccp_pgbouncer_databases* - SHOW DATABASES
- *ccp_pgbouncer_clients* - SHOW CLIENTS
- *ccp_pgbouncer_servers* - SHOW SERVERS
- *ccp_pgbouncer_lists* - SHOW LISTS

pg_stat_statements Metrics Collecting all per-query metrics into Prometheus could greatly increase storage requirements and heavily impact performance. Therefore, the metrics below give simplified numeric metrics on overall statistics and Top N queries. N is set as the LIMIT value in the `crunchy_pg_stat_statements_collector.yml` collections file. If you would like to adjust this number, it is recommended to make a copy of this collection file and use that modified file in your sql_exporter collector file config instead.

Note that the statistics for individual queries can only be reset on PG12+. Prior to that, pg_stat_statements must have all statistics reset to redo the top N queries.

- *ccp_pg_stat_statements_top_max_time_ms* - Maximum time spent in the statement in milliseconds per database/user/query for the top N queries
- *ccp_pg_stat_statements_top_mean_time_ms* - Average query runtime in milliseconds per database/user/query for the top N queries
- *ccp_pg_stat_statements_top_total_time_ms* - Total time spent in the statement in milliseconds per database/user/query for the top N queries
- *ccp_pg_stat_statements_total_calls_count* - Total number of queries run per user/database
- *ccp_pg_stat_statements_total_mean_time_ms* - Mean runtime of all queries per user/database
- *ccp_pg_stat_statements_total_row_count* - Total rows returned from all queries per user/database
- *ccp_pg_stat_statements_total_time_ms* - Total runtime of all queries per user/database

System

*NIX Operating System metrics (Linux, BSD, etc) are collected using the node_exporter provided by the Prometheus team. pgMonitor only collects the default metrics provided by node_exporter, but many additional metrics are available if needed.

Suggested Optional Metrics

There are many other suggestions, projects, and exporters out there that can provide additional metrics not included by default with pgMonitor. Some recommendations are below

https://docs.percona.com/pg-stat-monitor/ - Similar to pg_stat_statements, but provides deeper analysis on individual query statistics and performance. Note that this can greatly increase the metric storage requirements, but it can be extremely useful when trying to narrow down more complex query performance issues.

https://www.ansible.com/blog/red-hat-ansible-tower-monitoring-using-prometheus-node-exporter-grafana/ - Ansible Tower has a builtin exporter that can provide related metrics

Legacy postgres_exporter Setup

If you had been using pgMonitor prior to version 5.0.0, postgres_exporter was the method used to collect PostgreSQL metrics. This exporter can still be used with 5.0.0, but there are some additional steps required. You MUST migrate to sql_exporter ASAP as postgres_exporter has been deprecated and will be removed in the near future. Custom query support will be dropped upstream from postgres_exporter at some point in the future which will break pgMonitor as it relies solely on custom queries. No new features of pgMonitor are being developed around postgres_exporter.

Most of the installation steps are the same as above with the below differences for the relevant sections.

Available Packages

Package Name	Description
pgbouncer_fdw	Package for the pgbouncer_fdw extension. Only necessary when using postgres_exporter
pgmonitor-pg-common	Package containing postgres_exporter items common for all versions of PostgreSQL
pgmonitor-pg##-extras	Crunchy-optimized configurations for postgres_exporter. Note that each major version of PostgreSQL has its own extras package (pgmonitor-pg13-extras, pgmonitor-pg14-extras, etc).
postgres_exporter	Base package for postgres_exporter

Configuration File Installation The files contained in this repository are assumed to be installed in the following locations with the following names. In the instructions below, you should replace a double-hash (`##`) with the two-digit major version of PostgreSQL you are running (ex: 12, 13, 14, etc.).

pgMonitor Configuration File	System Location
postgres_exporter/common/setup.sql	/etc/postgres_exporter/##/setup.sql
postgres_exporter/common/pg##/queries*.yml	/etc/postgres_exporter/##/queries*.yml
postgres_exporter/common/queries*.yml	/etc/postgres_exporter/##/queries*.yml
postgres_exporter/linux/crontab.txt	/etc/postgres_exporter/##/crontab.txt
postgres_exporter/linux/crunchy-postgres-exporter@.service	/usr/lib/systemd/system/crunchy-postgres-exporter@.service
postgres_exporter/linux/pg##/sysconfig.postgres_exporter_pg##	/etc/sysconfig/postgres_exporter_pg##
postgres_exporter/linux/pg##/sysconfig.postgres_exporter_pg##_per_db	/etc/sysconfig/postgres_exporter_pg##_per_db
postgres_exporter/linux/queries_*.yml	/etc/postgres_exporter/##/queries_*.yml
postgres_exporter/linux/pgbackrest-info.sh	/usr/bin/pgbackrest-info.sh
postgres_exporter/linux/pgmonitor.conf	/etc/pgmonitor.conf (multi-backrest-repository/container environment only)

Service Configuration The following files contain defaults that should enable the exporters to run effectively on your system for the purposes of using pgMonitor. Please take some time to review them.

- {{< shell >}}/etc/sysconfig/postgres_exporter_pg##{{< /shell >}}
- {{< shell >}}/etc/sysconfig/postgres_exporter_pg##_per_db{{< /shell >}}

Note that {{< shell >}}/etc/sysconfig/postgres_exporter_pg##{{< /shell >}} & {{< shell >}}postgres_exporter_pg##_per_db{{< /shell >}} are the default sysconfig files for monitoring the database running on the local socket at /var/run/postgresql and connect to the "postgres" database. If you've installed the pgMonitor setup to a different database, modify these files accordingly or make new ones. If you make new ones, ensure the service name you enable references this file (see the Enable Services section below).

Monitoring Setup

Query File	Description
setup.sql	Creates `ccp_monitoring` role with all necessary grants. Creates all necessary database objects (functions, tables, etc) required for monitoring.
setup_metric_views.sql	Creates materialized views and maintenance objects for them. This feature is optional. See Materialized View Metrics.
queries_bloat.yml	postgres_exporter query file to allow bloat monitoring.
queries_global.yml	postgres_exporter query file with minimal recommended queries that are common across all PG versions and only need to be run once per database instance.

Query File	Description
queries_global_dbsize.yml	postgres_exporter query file that contains metrics for monitoring database size. This is a separate file to allow the option to use a materialized view for very large databases
queries_global_matview.yml	postgres_exporter query file that contains alternative metrics that use materialized views of common metrics across all PG versions
queries_per_db.yml	postgres_exporter query file with queries that gather per database stats. WARNING: If your database has many tables this can greatly increase the storage requirements for your prometheus database. If necessary, edit the query to only gather tables you are interested in statistics for. The "PostgreSQL Details" and the "CRUD Details" Dashboards use these statistics.
queries_per_db_matview.yml	postgres_exporter query files that contains alternative metrics that use materialized views of per database stats
queries_general.yml	postgres_exporter query file for queries that are specific to the version of PostgreSQL that is being monitored.
queries_backrest.yml	postgres_exporter query file for monitoring pgBackRest backup status. By default, new backrest data is only collected every 10 minutes to avoid excessive load when there are large backup lists. See sysconfig file for exporter service to adjust this throttling.
queries_pgbouncer.yml	postgres_exporter query file for monitoring pgbouncer.
queries_pg_stat_statements.yml	postgres_exporter query file for specific pg_stat_statements metrics that are most useful for monitoring and trending.

By default, there are two postgres_exporter services expected to be running. One connects to the default {{< shell >}}postgres{{< /shell >}} database that most PostgreSQL instances come with and is meant for collecting global metrics that are the same on all databases in the instance (connection/replication statistics, etc). This service uses the sysconfig file {{< shell >}}postgres_exporter_pg##{{< /shell >}}. Connect to this database and run the setup.sql script to install the required database objects for pgMonitor.

The second postgres_exporter service is used to collect per-database metrics and uses the sysconfig file {{< shell >}}postgres_exporter_pg##_per_db{{< /shell >}}. By default it is set to also connect to the {{< shell >}}postgres{{< /shell >}} database, but you can add as many additional connection strings to this service for each individual database that you want metrics for. Per-db metrics include things like table/index statistics and bloat. See the section below for monitorig multiple databases for how to do this.

Note that your pg_hba.conf will have to be configured to allow the {{< shell >}}ccp_monitoring{{< /shell >}} system user to connect as the {{< shell >}}ccp_monitoring{{< /shell >}} role to any database in the instance. As of version 4.0 of pg_monitor, the postgres_exporter service is set by default to connect via local socket, so passwordless local peer authentication is the expected default. If password-based authentication is required, we recommend using SCRAM authentication, which is supported as of version 0.7.x of postgres_exporter. See our blog post for more information on SCRAM - https://info.crunchydata.com/blog/how-to-upgrade-postgresql-passwords-to-scram

postgres_exporter only takes a single yaml file as an argument for custom queries, so this requires concatenating the relevant files together. The sysconfig files for the service help with this concatenation task and define the variable {{< yaml >}}QUERY_FILE_LIST{{< /yaml >}}. Set this variable to a space delimited list of the full path names to all files that contain queries you want to be in the single file that postgres_exporter uses.

For example, to use just the common queries for PostgreSQL 12 modify the relevant sysconfig file as follows:

```
QUERY_FILE_LIST="/etc/postgres_exporter/12/queries_global.yml
   /etc/postgres_exporter/12/queries_general.yml"
```

As an another example, to include queries for PostgreSQL 13 as well as pgBackRest, modify the relevant sysconfig file as follows:

```
QUERY_FILE_LIST="/etc/postgres_exporter/13/queries_global.yml
   /etc/postgres_exporter/13/queries_general.yml /etc/postgres_exporter/13/queries_backrest.yml"
```

For replica servers, the setup is the same except that the setup.sql file does not need to be run since writes cannot be done there and it was already run on the primary.

Materialized View Metrics With large databases/tables and some other conditions, certain metrics can cause excessive load. For those cases, materialized views and alternative metric queries have been made available. The materialized views are refreshed on their own schedule independent of the Prometheus data scrape, so any load that may be associated with gathering the underlying data is mitigated. A configuration table, seen below, contains options for how often these materialized views should be refreshed. And a single procedure can be called to refresh all materialized views relevant to monitoring.

For every database that will be collecting materialized view metrics, you will have to run the {{< shell >}}setup_metric_views.sql{{< /shell >}} file against that database. This will likely need to be run as a superuser and must be run after running the base setup file mentioned above to create the necessary monitoring user first.

```
psql -U postgres -d alphadb -f setup_metric_views.sql
psql -U postgres -d betadb -f setup_metric_views.sql
```

The {{< shell >}}/etc/postgres_exporter/##/crontab.txt{{< /shell >}} file has an example entry for how to call the refresh procedure. You should modify this to run as often as you need depending on how recent you need your metric data to be. This procedure is safe to run on the primary or replicas and will safely exit if the database is in recovery mode.

Configuration table {{< shell >}}monitor.metric_views{{< /shell >}}:

Column	Description
view_schema	Schema containing the materialized view
view_name	Name of the materialized view
concurrent_refresh	Boolean that sets whether this materialized view can be refreshed concurrently (requires a unique index)
run_interval	How often this materialized view should have its data refreshed. Must be a value compatible with the PG interval type
last_run	Timestamp of the last time this view was refreshed
active	Boolean that sets whether this view should be refreshed when the procedure is called
scope	Whether the data contained in the view is per-database or instance-wide. Currently unused

You are also free to use this materialized view system for your own custom metrics as well. Simply make a materialized view, add its name to the configuration table and ensure the user running the refresh has permissions to do so for your view(s).

PGBouncer In order to monitor pgbouncer with postgres_exporter, the pgbouncer_fdw maintained by CrunchyData is required. Please see its repository for full installation instructions. A package for this is available for Crunchy Data customers.

https://github.com/CrunchyData/pgbouncer_fdw

Once that is working, you should be able to add the {{< shell >}}queries_pgbouncer.yml{{< /shell >}} file to the {{< yaml >}}QUERY_FILE_LIST{{< /shell >}} for the exporter that is monitoring the database where the FDW was installed.

Enable Services To most easily allow the use of multiple postgres exporters, running multiple major versions of PostgreSQL, and to avoid maintaining many similar service files, a systemd template service file is used. The name of the sysconfig EnvironmentFile to be used by the service is passed as the value after the "@" and before ".service" in the service name. The default exporter's sysconfig file is named "postgres_exporter_pg##" and tied to the major version of postgres that it was installed for. A similar EnvironmentFile exists for the per-db service. Be sure to replace the ## in the below commands first!

```
sudo systemctl enable crunchy-postgres-exporter@postgres_exporter_pg##
sudo systemctl start crunchy-postgres-exporter@postgres_exporter_pg##
sudo systemctl status crunchy-postgres-exporter@postgres_exporter_pg##

sudo systemctl enable crunchy-postgres-exporter@postgres_exporter_pg##_per_db
sudo systemctl start crunchy-postgres-exporter@postgres_exporter_pg##_per_db
sudo systemctl status crunchy-postgres-exporter@postgres_exporter_pg##_per_db
```

Monitoring multiple databases and/or running multiple postgres exporters (RHEL) Certain metrics are not cluster-wide, so multiple exporters must be run to avoid duplication when monitoring multiple databases in a single PostgreSQL instance. To collect these per-database metrics, an additional exporter service is required and pgMonitor provides this using the following query file: ({{< shell >}}queries_per_db.yml{{< /shell >}}). In Prometheus, you can then define the global and per-db exporter targets for a single job. This will place all the metrics that are collected for a single database instance together.

{{< note >}}The "setup.sql" file does not need to be run on these additional databases if using the queries that pgMonitor comes with.{{< /note >}}

pgMonitor provides and recommends an example sysconfig file for this per-db exporter: {{< shell >}}sysconfig.postgres_exporter_pg##_pe /shell >}}. If you'd like to create additional exporter services for different query files, just copy the existing ones and modify the relevant lines, mainly the port, database name, and query file. The below example shows connecting to three databases in the same instance to collect their per-db metrics: `postgres`, `mydb1`, and `mydb2`.

```
OPT="--web.listen-address=0.0.0.0:9188
   --extend.query-path=/etc/postgres_exporter/14/queries_per_db.yml"
DATA_SOURCE_NAME="postgresql:///postgres?host=/var/run/postgresql/&user=ccp_monitoring&sslmode=disabl
```

As was done with the exporter service that is collecting the global metrics, also modify the {{< yaml >}}QUERY_LIST_FILE{{< /yaml >}} in the new sysconfig file to only collect per-db metrics

```
QUERY_FILE_LIST="/etc/postgres_exporter/14/queries_per_db.yml"
```

Since a systemd template is used for the postgres_exporter services, all you need to do is pass the sysconfig file name as part of the new service name.

```
sudo systemctl enable crunchy-postgres-exporter@postgres_exporter_pg14_per_db
sudo systemctl start cruncy-postgres-exporter@postgres_exporter_pg14_per_db
sudo systemctl status crunchy-postgres-exporter@postgres_exporter_pg14_per_db
```

Lastly, update the Prometheus auto.d target file to include the new exporter in the same job you already had running for this system

General Metrics *pg_up* - Database is up and connectable by metric collector. This metric is only available with postgres_exporter

Prometheus can be set up on any Linux-based system, but pgMonitor currently only supports running it on RHEL/CentOS 7 or later.

- Installation
  - Package Install
  - Non-Package Install
- Upgrading
- Setup
  - RHEL

Installation

Package Install

There are RPM packages available to Crunchy Data customers through the Crunchy Customer Portal. To access the pgMonitor packages, please follow the same instructions for setting up access to the Crunchy Postgres packages.

After installing via these RPMs, you can continue reading at the Setup section.

Available Packages

Package Name	Description
alertmanager	Base package for the Alertmanager
prometheus2	Base package for Prometheus 2.x
pgmonitor-alertmanager-extras	Custom Crunchy configurations for Alertmanager
pgmonitor-prometheus-extras	Custom Crunchy configurations for Prometheus

Non-Package Install

For installations without using packages provided by Crunchy Data, we recommend using the repository maintained at https://github.com/les rpm. Instructions for setup and installation are contained there. Note this only sets up the base service. The additional files and steps for pgMonitor still need to be set up as instructed below.

Or you can also download Prometheus and Alertmanager from the original site at https://prometheus.io/download. Note that no base service setup is provided here, just the binaries.

Minimum Versions pgMonitor has been tested with the following versions at a minimum. Later versions should generally work. If they do not, please open an issue on our Github.

- Prometheus 2.49.1
- Alertmanager 0.26.0

User and Configuration Directory Installation You will need to create a system user named {{< shell >}}ccp_monitoring{{< /shell >}} which you can do with the following command:

```
sudo useradd -m -d /var/lib/ccp_monitoring ccp_monitoring
```

Configuration File Installation The files contained in this repository are assumed to be installed in the following locations with the following names:

Prometheus The Prometheus data directory should be {{< shell >}}/var/lib/ccp_monitoring/prometheus{{< /shell >}} and owned by the {{< shell >}}ccp_monitoring{{< /shell >}} user. You can set it up with:

```
sudo install -d -m 0700 -u ccp_monitoring -g ccp_monitoring /var/lib/ccp_monitoring/prometheus
```

The following pgmonitor configuration files should be placed according to the following mapping:

pgMonitor Configuration File	System Location
prometheus/linux/crunchy-prometheus-service-rhel.conf	/etc/systemd/system/prometheus.service.d/crunch prometheus-service-rhel.conf
prometheus/linux/sysconfig.prometheus	/etc/sysconfig/prometheus
prometheus/linux/crunchy-prometheus.yml	/etc/prometheus/crunchy-prometheus.yml
prometheus/linux/auto.d/*.yml.example	/etc/prometheus/auto.d/*.yml.example
prometheus/linux/alert-rules.d/crunchy-alert-rules*.yml.example	/etc/prometheus/alert-rules.d/crunchy-alert-rules-*.yml.example
prometheus/common/auto.d/*.yml.example	/etc/prometheus/auto.d/*.yml.example
prometheus/common/alert-rules.d/crunchy-alert-rules*.yml.example	/etc/prometheus/alert-rules.d/crunchy-alert-rules-*.yml.example

Alertmanager The Alertmanager data directory should be `/var/lib/ccp_monitoring/alertmanager` and owned by the `ccp_monitoring` user. You can set it up with:

```
sudo install -d -m 0700 -o ccp_monitoring -g ccp_monitoring /var/lib/ccp_monitoring/alertmanager
```

The following pgMonitor configuration files should be placed according to the following mapping:

pgMonitor Configuration File	System Location
alertmanager/linux/crunchy-alertmanager-service-rhel.conf	/etc/systemd/system/alertmanager.service.d/crun alertmanager-service-rhel.conf
alertmanager/linux/sysconfig.alertmanager	/etc/sysconfig/alertmanager
alertmanager/common/crunchy-alertmanager.yml	/etc/prometheus/crunchy-alertmanager.yml

Upgrading

- If you are upgrading to version 5.0 and transitioning to using the new sql_exporter, please see the documentation in Upgrading to pgMonitor v5.0.0
- See the CHANGELOG for full details on both major & minor version upgrades.
- Note, items like the alert rules and configuration files often require user edits. The packages will install newer versions of these files, but if the user has changed their contents but kept the same file name, the package will not overwrite them. Instead it will make a file with an {{< shell >}}*.rpmnew{{< /shell >}} extension that contains the newer version of the file. These new files can be reviewed/compared to he user's file to incorporate any desired changes.

Setup

Setup on RHEL

Service Configuration The following files contain defaults that should enable Prometheus and Alertmanager to run effectively on your system for the purposes of using pgmonitor. You should take some time to review them.

If you need to modify them, see the notes in the files for more details and recommendations:

- {{< shell >}}/etc/systemd/system/prometheus.service.d/crunchy-prometheus-service-rhel.conf{{< /shell >}}
- {{< shell >}}/etc/systemd/system/alertmanager.service.d/crunchy-alertmanager-service-rhel.conf{{< /shell >}}

The below files contain startup properties for Prometheues and Alertmanager. Please review and modify these files as you see fit:

- {{< shell >}}/etc/sysconfig/prometheus{{< /shell >}}
- {{< shell >}}/etc/sysconfig/alertmanager{{< /shell >}}

The below files dictate how Prometheus and Alertmanager will behave at runtime for the purposes of using pgMonitor. Please review each file below and follow the instructions in order to set things up:

File	Instructions
/etc/prometheus/crunchy-prometheus.yml	Main configuration file for prometheus to set things like scrape intervals and alerting. blackbox_exporter monitoring can also be enabled if desired. Service file provided by pgMonitor expects config file to be named "crunchy-prometheus.yml". For full configuration options please see the Prometheus upstream documentation
/etc/prometheus/crunchy-alertmanager.yml	Setup alert target (e.g., SMTP, SMS, etc.), receiver and route information. Service file provided by pgMonitor expects config file to be named "crunchy-alertmanager.yml". For full configuration options please see the Alertmanager upstream documentation
/etc/prometheus/alert-ruled.d/crunchy-alert-rules-*.yml.example	Update rules as needed and remove ".example" suffix. Prometheus config provided by pgmonitor expects ".yml" files to be located in "/etc/prometheus/alert-rules.d/". Additional information on configuring alert rules can be found in the alert rules upstream documentation.
/etc/prometheus/auto.d/*.yml	You will need at least one file with a final ".yml" extension. Copy the example files to create as many additional targets as needed. Ensure the configuration files you want to use do not end in ".yml.example" but only with ".yml". Note that in order to use the provided Grafana dashboards, the extra "exp_type" label must be applied to all targets and be set appropriately (pg, node, etcd, pgbouncer, etc). Also, PostgreSQL targets make use of the "cluster_name" variable and should be given a relevant value so all systems (primary & replicas) can be related to each other when needed (Grafana dashboards, etc). See the example target files provided for how to set the labels for postgres or node exporter targets.

Blackbox Exporter By default, the Blackbox exporter probes are commented out in the {{< shell >}}crunchy-prometheus.yml{{< /shell >}} file; please see the notes in that commented out section. For the default IPv4 TCP port targets that pgMonitor configures the blackbox_exporter with, the desired monitoring targets can be configured under the {{< yaml >}}static_configs: targets{{< /yaml >}} section of the {{< yaml >}}blackbox_tcp_services{{< /yaml >}} job; some examples for Grafana & Patroni are given there. It is also possible to create another auto-scrape target directory similar to {{< shell >}}auto.d{{< /shell >}} and manage your blackbox targets more dynamically.

If you configure additional probes beyond the one that pgMonitor comes with, you will need to create a different Prometheus {{< yaml

>}}job_name{{< /yaml >}} for them for the given {{< yaml >}}params: module{{< /yaml >}} name.

An example rules file for monitoring Blackbox probes, {{< shell >}}crunchy-alert-rules-blackbox.yml.example{{< /shell >}}, is available in the {{< shell >}}alert-rules.d{{< /shell >}} folder.

Enable Services To enable and start Prometheus as a service, execute the following commands:

```
sudo systemctl enable prometheus
sudo systemctl start prometheus
sudo systemctl status prometheus
```

To enable and start Alertmanager as a service, execute the following commands:

```
sudo systemctl enable alertmanager
sudo systemctl start alertmanager
sudo systemctl status alertmanager
```


Included Dashboards

pgMonitor comes with several dashboards ready to be used with automatic provisioning. They provide examples of using the metrics from the postgres_exporter and node_exporter. Since provisioned dashboards cannot be edited directly in the web interface, if any custom changes are desired, it is recommended to make a copy of them and make your changes there.

Filename	Dashboard Name	Description
etcd/ETCD_Details.json	ETCD Details	Provides details on the status of the ETCD cluster monitored by pgMonitor.
linux/Filesystem_Details.json	Filesystem Details	Provides details on the filesystem metrics (disk usage, IO, etc).
linux/Network_Details.json	Network Details	Provides details on network usage (utilization, traffic in/out, netstat, etc).
linux/OS_Details.json	OS Details	Provides details on operating system metrics (cpu, memory, swap, disk usage). Links to Filesystem Details dashboard.
linux/OS_Overview.json	OS Overview	Provides an overview that shows the up status of each system monitored by pgMonitor.
postgres/Bloat_Details.json	Bloat Details	Provides details on database bloat (wasted space). Provides overview and top-n statistics.
postgres/CRUD_Details.json	CRUD Details	Provides details on Create, Read, Update, Delete (CRUD) statistics on a per-table basis.
postgres/PGBackrest.json	pgBackRest	Provides details on backups performed with pgBackRest. Also provides recovery window to show timeframe available for PITR.
postgres/PG_Details.json	PostgreSQL Details	Provides detailed information for each PostgreSQL instance (connections, replication, wraparound, etc).
postgres/postgres_exporter/PG_Overview_postgres_exporter.json	PostgreSQL Overview	Provides an overview of all PostgreSQL systems being monitored when the postgres_exporter is being used. Indicates whether a system is a Primary or Replica. Can click on each panel to open up the PostgreSQL Details for that system.
postgres/sql_exporter/PG_Overview_sql_exporter.json	PostgreSQL Overview (sql_exporter)	Provides an overview of all PostgreSQL systems being monitored when the sql_exporter is being used. Indicates whether a system is a Primary or Replica. Can click on each panel to open up the PostgreSQL Details for that system.
postgres/QueryStatistics.json	Query Statistics	Provides an overview of statistics collected by the pg_stat_statements extension.

Filename	Dashboard Name	Description
postgres/TableSize_Details.json	TableSize Details	Provides size details on a per-table basis for the given database.
pgbouncer/direct/PGBouncer.json	PgBouncer (direct)	Provides details from the PgBouncer statistics views when connecting directly to PgBouncer with the sql_exporter.
pgbouncer/fdw/PGBouncer.json	PgBouncer	Provides details from the PgBouncer statistics views when using the pgbouncer_fdw.
prometheus/Prometheus_Alerts.json	Prometheus Alertsi	Provides a summary list of current and recent alerts that have fired in Prometheus.

Installation

Package Install

There are RPM packages available to Crunchy Data customers through the Crunchy Customer Portal. To access the pgMonitor packages, please follow the same instructions for setting up access to the Crunchy Postgres packages.

If you install the below available packages, you can continue reading at the Setup section. Dashboards have been split up into different packages to allow users to only have dashboards that are relevant to their environment appear in the provisioned dashboards location.

Available Packages

Package Name	Description
grafana	Base package for grafana
pgmonitor-grafana-extras-common	Crunchy files common for all Grafana installations (Ex. datasource & dashboard provisioning)
pgmonitor-grafana-extras-etcd	Crunchy dashboards for etcd (etcd built-in exporter)
pgmonitor-grafana-extras-linux	Crunchy dashboards for Linux OS metrics (node_exporter)
pgmonitor-grafana-extras-haproxy	Crunchy dashboards for HAProxy (built-in exporter)
pgmonitor-grafana-extras-pg	Crunchy dashboards for PostgreSQL metrics common to all PG exporters
pgmonitor-grafana-extras-pg-postgres-exporter	Crunchy dashboards for metrics provided by postgres_exporter (see version 5 upgrade for deprecation notice)
pgmonitor-grafana-extras-pg-sql-exporter	Crunchy dashboards for metrics provided by the sql_exporter
pgmonitor-grafana-extras-pgbouncer-direct	Crunchy dashboards for PgBouncer metrics collected directly from bouncer via the sql_exporter
pgmonitor-grafana-extras-pgbouncer-fdw	Crunchy dashboards for PgBouncer metrics collected via the pgbouncer_fdw
pgmonitor-grafana-extras-prometheus	Crunchy dashboards for proividing direct Prometheus visualization (alerting)

Non-Package Install

Create the following directories on your grafana server if they don't exist:

```
mkdir -p /etc/grafana/provisioning/{datasources,dashboards}
mkdir -p /etc/grafana/crunchy_dashboards
```

pgmonitor Configuration File	System Location
grafana/crunchy_grafana_datasource.yml	/etc/grafana/provisioning/datasources/datasource.yml
grafana/crunchy_grafana_dashboards.yml	/etc/grafana/provisioning/dashboards/dashboards.yml

Review the {{< shell >}}crunchy_grafana_datasource.yml{{< /shell >}}} file to ensure it is looking at your Prometheus database. The included file assumes Grafana, Prometheus, and Alertmanager are running on the same system. DO NOT CHANGE the data-

source {{< yaml >}}uid{{< /yaml >}} or {{< yaml >}}name{{< /yaml >}} fields if you will be using the dashboards provided in this repo. They assume those values and will not work otherwise. Any other options can be changed as needed. Save the {{< shell >}}crunchy_grafana_datasource.yml{{< /shell >}} file and rename it to {{< shell >}}/etc/grafana/provisioning/datasources/datasources.yml{{< /shell >}}. Restart grafana and confirm through the web interface that the datasource was provisioned and working.

Review the {{< shell >}}crunchy_grafana_dashboards.yml{{< /shell >}} file to ensure it's looking at where you stored the provided dashboards. By default it is looking in {{< shell >}}/etc/grafana/crunchy_dashboards{{< /shell >}}. Save this file and rename it to {{< shell >}}/etc/grafana/provisioning/dashboards/dashboards.yml{{< /shell >}}. Restart grafana so it picks up the new config.

Save all of the desired .json dashboard files (see table above) to the {{< shell >}}/etc/grafana/crunchy_dashboards{{< /shell >}} folder. All of them are not required, so if there is a dashboard you do not need, it can be left out.

Please note that due to the change from postgres_exporter to sql_exporter, and its ability to connect directly to pgBouncer to collect its metrics, some dashboards are specific to one exporter or the other. Please use the relevant dashboards accordingly based on their descriptions above.

Upgrading

- If you are upgrading to version 5.0 and transitioning to using the new sql_exporter, please see the documentation in Upgrading to pgMonitor v5.0.0
- See the CHANGELOG for full details on both major & minor version upgrades.
- Note that if you are using the included dashboards that are managed via the provisioning system, they will automatically be updated. If you've made any changes to configuration files and kept their default names, the package will not overwrite them and will instead make a new file with an {{< shell >}}*.rpmnew{{< /shell >}} extension. You can compare your file and the new one and incorporate any changes as needed or desired.

Setup

Setup on RHEL

Configuration Database By default Grafana uses an SQLite database to store configuration and dashboard information. We recommend using a PostgreSQL database for better long term scalability. Before doing any further configuration, including changing the default admin password, set the `grafana.ini` to point to a postgresql instance that has a database created for it.

In psql run the following:

```
    CREATE ROLE grafana WITH LOGIN;
    CREATE DATABASE grafana;
    ALTER DATABASE grafana OWNER TO grafana;
    \password grafana
```

You may also need to adjust your `pg_hba.conf` to allow grafana to connect to your database.

In your `grafana.ini`, set the following options at a minimum with relevant values:

```
[database]

type = postgres
host = 127.0.0.1:5432
name = grafana
user = grafana
password = """mypassword"""
```

Now enable and start the grafana service

```
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
sudo systemctl status grafana-server
```

Navigate to the web interface: https://<ip-address>:3000. Log in with admin/admin (be sure to change the admin password) and check settings to ensure the postgres options have been set and are working.

Datasource & Dashboard Provisioning

Grafana provides the ability to automatically provision datasources and dashboards via configuration files instead of having to manually import them either through the web interface or the API. Note that provisioned dashboards cannot be directly edited and saved via the web interface. See the Grafana documentation for how to edit/save provisioned dashboards: http://docs.grafana.org/administration/provisioning/#making-changes-to-a-provisioned-dashboard. If you'd like to customize these dashboards, we recommend first adding them via provisioning then saving them with a new name. You can then either manage them via the web interface or add them to the provisioning system.

The extras package takes care of putting all these files in place. If you did not use the Crunchy package to install Grafana, see the additional instructions above. Once that is done, the only additional setup that needs to be done is to set the "provisioning" option in the `grafana.ini` to point to the top level directory if it hasn't been done already.

```
[paths]
provisioning = /etc/grafana/provisioning
```

v5.3.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 5.3.0. This release brings several important bugfixes and updates dashboards in Grafana.

Breaking Changes / Porting Guide

- sql_exporter - Dropped support for PostgreSQL 11 and 12

Minor Changes

- grafana -Add pgBouncer dashboard to containers folder
- grafana -Updated containers dashboards to use latest ccp metrics in v5.2.1
- grafana -Updated containers dashboards/alerts to allow OTel or postgres-exporter values
- grafana - Update HAProxy dashboard to the latest one from upstream (Revision 11)
- prometheus - Add requested PGNoPrimary and PGNoReplica alerts for containers
- prometheus - Moved the ExporterDown alert to its own common alerts file and have it be enabled by default (no .example extension on the file name)
- prometheus - Remove unnecessary absence alerts. The general ExporterDown metric can cover these scenarios
- sql_exporter - Consolidated setup file from different PostgreSQL versions to a single common file

Bugfixes

- grafana - Fix PG Overview dashboard for sql_exporter so that links to the PG Details dashboard have the proper, relevant node already chosen
- sql_exporter - Removed the pg_settings_checksum metric since it was not working properly. Requires new version of pgmonitor-extension.

v5.2.1

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 5.2.1. This release brings support for monitoring the latest PgBouncer version 1.24.

Bugfixes

- sql_exporter - Create new collections file for pgBouncer 1.24+

v5.2.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 5.2.0. This release brings support for the latest pgmonitor extension version 2.1.0 and many bug fixes.

Major Changes

- sql_exporter - Updated metric queries for pgmonitor-extension 2.1.0

Minor Changes

- grafana - Improve Query Statistics dashboard to more accurately report lifetime vs time range statistics
- grafana - Removed the `Last Backup Size (Total)` panel from the pgBackRest Grafana dashboard since it was backed by the removed metric.
- postgres_exporter - Removed the metric `ccp_backrest_last_info_repo_total_size_bytes`.
- prometheus - Added new alerts to monitor the new metrics for the minimum required version of the pgmonitor extension
- sql_exporter - Add additional metrics for monitoring replication slot status. For PG16+ monitor for conflicts. For PG17+, monitor synced and failover status.

- sql_exporter - Add scrape_error_drop_interval setting to the configuration example
- sql_exporter - Added new metrics to monitor pgmonitor-extension version. - `ccp_pgmonitor_extension_global_version` - shows the currently installed version of the extension on the global database as an integer - `ccp_pgmonitor_extension_per_db_version` - shows the currently installed version of the extension on each monitored user database as an integer - `ccp_pgmonitor_extension_globa` - shows whether the currently installed version of the extension is the minimum required for this version of pgMonitor on the global database (0 true, 1 false) - `ccp_pgmonitor_extension_per_db_min_version_installed` - shows whether the currently installed version of the extension is the minimum required for this version of pgMonitor on each monitored user database (0 true, 1 false)
- sql_exporter - Removed the metric `ccp_backrest_last_info_repo_total_size_bytes`. When block incremental backups are enabled, this metric is no longer available from pgBackRest.

Bugfixes

- grafana - Fix Cache Hit Ratio panel on PG Details dashboard to always be lines. Depending on data returned was sometimes being shown as points.
- grafana - Fix pgBackRest recovery window panel showing multiple values after a PostgreSQL switchover
- postgres_exporter - Disable all collectors included with postgres_exporter by default in example configuration. The other options to disable default metrics are not applied to the new collections.
- postgres_exporter - Fix the `ccp_table_size_size_bytes` metric to remove the duplicate word and just be `ccp_table_size_bytes`
- sql_exporter - Fix the `ccp_table_size_size_bytes` metric to remove the duplicate word and just be `ccp_table_size_bytes`
- sql_exporter - Fix the names of ccp_pgbouncer_database_db_conn_perc_used, ccp_pgbouncer_database_paused, ccp_pgbouncer_ and ccp_pgbouncer_list_item_count to be consistent with the old metric names from postgres exporter. These new names are the ones expected in the Grafana dashboard so this change fixes that to work properly again.
- sql_exporter - Removed extraneous double quote at the end of the pgbouncer fdw collector file

v4.12.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 4.12.0. This release primarily brings support for Grafana 10.4. See Changelog for additional information.

Breaking Changes / Porting Guide

- grafana - Update the dashboards to support Grafana 10.4 so that we're on an officially supported version of Grafana. This does potentially break backward compatibility with Grafana 9.x, so an update to Grafana 10.4 will be required with this version of pgMonitor.

Bugfixes

- grafana - Fix etcd dashboard to use new metric names in etcd 3.5
- postgres_exporter - Fix query for database table size to remove duplicate word
- postgres_exporter - Fix query for pgBackRest monitoring to handle 3 number versionso

v5.1.1

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 5.1.1. This release brings additional support for PostgreSQL 17 to postgres_exporter to allow easier migration to sql_exporter.

Minor Changes

- postgres_exporter - Files to help transition off of postgres_exporter.

v5.1.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 5.1.0. This release brings support for PostgreSQL 17. It also brings more flexible Grafana dashboards, an HAProxy dashboard and better support for etcd and pgBackRest.

Major Changes

- grafana - add support for new metrics in PG17
- sql_exporter - add support for PG17

Minor Changes

- grafana - Add a new variable dropdown to all dashboards for the Datasource. Allows more flexibility when importing the dashboard to different environments.
- grafana - Add panel to PG Details dashboard to track autovac workers running vs max
- grafana - add a dashboard for HAProxy
- sql_exporter - Add metrics to track current autovacuum workers running and max autovacuum workers
- sql_exporter - A password for the ccp_monitoring database role is no longer set when using the setup_db.sql file.
- sql_exporter - Make the default privileges for the setup_db.yml file world readable (when installing via package).

Bugfixes

- grafana - Fix etcd dashboard to use new metric names in etcd 3.5
- postgres_exporter - Fix query for pgBackRest monitoring to handle 3 number versions

5.0.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 5.0.0. This release brings support for a new Prometheus exporter for PostgreSQL - sql_exporter. It also supports a new monitoring extension to make metric collection easier and more performant. This changelog contains all changes that have been added since the 4.11.0 release.

Please see the 5.0.0 upgrade documentation for more information on converting to the new sql_exporter.

Major Changes

- grafana - Add new dashboards for sql_exporter support. New PostgreSQL Overview and PgBouncer direct metrics dashboards
- grafana - New Grafana minimum version is now 10.4. All dashboards have been updated to fix AngularJS deprecation warnings and re-exported from 10.4.
- grafana - Organize packages to allow better choice of available Grafana dashboards
- grafana - Remove top level general Overview dashboard
- pgmonitor-extension - Add more extensive support for materialized views and refreshed tables for expensive or custom metric queries
- pgmonitor-extension - Add support for using the PostgreSQL pgmonitor-extension to aid in metrics collection with sql_exporter
- postgres_exporter - Note that postgres_exporter is still supported but will be deprecated in a future version
- sql_exporter - Add support for directly connecting to PgBouncer to collect metrics
- sql_exporter - Add support for new PostgreSQL metrics collecting exporter (sql_exporter)

Minor Changes

- prometheus - Added OOMKiller Alert using node_exporter metrics

Bugfixes

- docs - add reference links to upstream configuration docs
- exporter - fix the pgbackrest-info.sh to force the necessary console output level that it expects
- grafana - fix some queries that were searching on the wrong label (datname vs. dbname)
- sql_exporter - add new metric for n_tup_newpage_upd
- sql_exporter - use the new views from pgmonitor-extension instead of full queries

4.11.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 4.11.0. This release primarily updates support for the underlying applications to more recent versions. This changelog contains all changes that have been added since the 4.10.0 release.

Minor Changes

- alertmanager - minimum version 0.23, maximum 0.26.x
- blackbox_exporter - minimum version 0.22.x, maximum 0.24.x
- grafana - minimum version 9.2.19, maximum 9.9.x
- node_exporter - minimum version 1.5.0, maximum 1.7.x
- postgres_exporter - minimum version 0.10.1, maximum 0.15.x
- prometheus - minimum version 2.38, maximum 2.49.x

4.10.0

Release Summary

Crunchy Data is pleased to announce the availability of pgMonitor 4.10.0. This release primarily adds support for PostgreSQL 16. This changelog contains all changes that have been added since the 4.9.0 release.

Minor Changes

- postgres_exporter - Add support for PostgreSQL 16
- containers - The datasource for containers is named PROMETHEUS. Update dashboards to use the hardcoded name.
- grafana - Adjust the cache hit graph to do a 1m rate vs lifetime ratio
- grafana - Relabel the cache hit ratio dial properly mark it as the lifetime cache hit replication
-

4.9.0

Release Summary

Version 4.9.0 of pgMonitor includes updates to add additional metrics and now better supports monitoring multiple PgBouncer hosts. Please see the full CHANGELOG for additional information about this release.

Major Changes

- postgres_exporter - Added options for using materialized views to collect metrics that may cause longer query runtimes (object sizing, statistics, etc)
- postgres_exporter - Moved the database size metric out of the 'queries_global.yml' file and into the 'queries_global_dbsize.yml' file to allow an optional materialized view metric. Ensure query file configuration list is updated to account for this change

Minor Changes

- blackbox_exporter - added additional probe for TCP with TLS enabled
- grafana - Add panel to Query Statistics dashboard for top WAL stats by bytes
- grafana - Minimum version of Grafana is now 9.2.19
- grafana - Update dashboard to support multiple PgBouncer targets exported by new pgbouncer_fdw
- postgres_exporter - Add WAL statistics for pg_stat_statements
- postgres_exporter - Filter out idle-in-transaction sessions from general max query runtime metrics.
- postgres_exporter - Update query file to support PgBouncer_fdw 1.0.0+
- prometheus - Add alert for cases where a PostgreSQL cluster does not have an instance that is the leader/primary
- prometheus - Allow node_exporter's load alert to be based on the CPU count. Allows lowering of default thresholds and more accurate alerting
- prometheus - Enable the PGDataChecksum alert by default for PG12+
- prometheus - Update the example files to provide better guidance on proper configuration
- prometheus - added additional job example to scan TCP probes with TLS

Bugfixes

- grafana - fixed dashboard links that broke when Grafana removed support for the `/dashboard/db/:slug` endpoint in v8

4.8.0

Release Summary

Version 4.8.0 of pgMonitor includes support for PostgreSQL 15. Please see the CHANGELOG for additional information about this release.

NOTE: PostgreSQL 9.6 and 10 official support has been dropped as of this version

Major Changes

- Update to support PostgreSQL 15 (https://github.com/CrunchyData/pgmonitor/issues/296)

Minor Changes

- Disable JIT for the ccp_monitoring user to avoid memory leak issues (https://github.com/CrunchyData/pgmonitor/issues/295)
- Update prometheus sysconfig file to use up to date startup values (https://github.com/CrunchyData/pgmonitor/issues/293)

Bugfixes

- Fixed pgbackrest-info.sh script to account for old default pgBackRest config file not existing
- Remove unnecessary $-escaping in the service file (https://github.com/CrunchyData/pgmonitor/issues/301)
- Update global sysconfig file to have proper general queries file (https://github.com/CrunchyData/pgmonitor/issues/297)

4.7

NOTE: This is the last version of pgMonitor that will contain support for PostgreSQL 9.6.

New Features

- Added metric to monitor for errors encountered in pgBackRest (ccp_backrest_last_info_backup_error). Also added example Prometheus alert.
- New, more detailed etcd dashboard for Grafana
- Added Prometheus alerts for monitoring Patroni and etcd

Bug Fixes

- Fixed inconsistency in the OS Details Grafana dashboard between the small left panel for the filesystem and the other filesystem panels.
- Fixed postgres_exporter queries for PgBouncer to select the proper "user" column.

4.6

New Features

- Support for pgBackRest multiple repositories in metrics and Grafana dashboards. Minimum requirement of pgBackRest is now 2.33.

Bug Fixes

- Allow PostgreSQL Overview Grafana dashboard to work with Grafana 8.x
- Fix queries to use clock_timestamp() vs now() to avoid negative values in some query results
- Fix postgres_exporter sysconfig to properly support multiple options that have .yml files as values (allows tls support).

4.5

New Features

- Add preliminary support for PostgreSQL 14
- Minimum required version of Grafana has been updated to 7.5.x
- Updated Grafana Overview dashboards to support new Stat panel
- Updated PostgreSQL Details Grafana dashboard with more information and to be able to present data grouped by clusters. The pgBackRest panel was removed from this dashboard.
- The pgBackRest Grafana dashboard now presents data on a per-stanza basis
- Removed deprecated node_exporter metrics from Grafana OS Details dashboard. Reorganized panels.
- Added a basic Network Activity dashboard to Grafana using default metrics that come with node_exporter.
- The pgMonitor repository has been reorganized around which platforms files apply to. Some files have also been renamed as part of this reorganization.
- Extended the default alert threshold for pgBackRest backups to give a buffer time and avoid false positives when backup runtimes vary.
- Added a default alert for PostgreSQL failover that should work in any scenario to produce an alert when the recovery status of a PostgreSQL database changes (replica -> primary or primary -> replica). Note that this alert will auto-resolve after 5 minutes (by default) since it is just looking for recent state changes. The alert is meant to be acted upon immediately to see what may have occurred on the systems involved.
- Added metric to monitor and alert on blocked queries

Bug Fixes

- Fixed several incorrect metric names in alert expressions for the example alert files. Please review all alerts to ensure your expressions are checking the correct metrics, making special note of the following:
  - PGSettingsChecksum
  - PGDBSize
  - PGReplicationByteLag
  - MemoryAvailable
  - SwapUsage
  - All pgBackRest alerts

- Fixed pgBackRest metrics not reporting all backups in all stanzas for a given repository in some configuration setups. Each database will now only report back monitoring for the stanzas that are part of its own instance. Previously all database instances reported back all stanzas in the target repository.
- Fixed incorrect title of panel on Grafana PostgreSQL Details dashboard from "Transactions Per Minute" to "Transactions Per Second".

Manual Intervention Changes

- pgBackRest monitoring has been expanded to better support more configuration layouts to address the above bug fix. The pgbackrest-info.sh script has been updated as part of this and this also requires re-running the setup SQL script to update the monitoring function within the database. Note again that the setup script name has changed from "setup_pg11.sql" to "setup.sql", so be sure you are running the setup script from the properly versioned folder.
- For the PostgreSQL Grafana dashboards to be able to choose data to present on a per-cluster basis, a new custom label (`cluster_name`) must be added to all PostgreSQL targets in Prometheus. Note that this change will cause all PostgreSQL metrics to change colors from the point of the change forward. Also when displaying a time period before and after this change, duplicated Legend items may appear.

4.4-1

New Features

Bug Fixes

- Limit SQL function's search_path to predefined list of schemas

Non-backward Compatible Changes

Manual Intervention Changes

4.4

New Features

- Added support for PostgreSQL 13
- Added queries and dashboards for pgnodemx/container support
- Added metrics and Grafana dashboard for pg_stat_statements
- Added metrics for monitoring longest blocked query time

Bug Fixes

Non-backward Compatible Changes

Manual Intervention Changes

- To add pg_stat_statements metrics to an existing installation you will need to do the following:
  - Add the relevant queries_pg_stat_statements_pg##.yml file to the QUERY_FILE_LIST in the exporter sysconfig file
  - Add a PG_STAT_STATEMENTS_LIMIT line to the exporter sysconfig file with a desired limit for the top N queries. Default for a new install is 20.

4.3

New Features

Bug Fixes

- Fixed syntax error in example prometheus alert rules file for postgresql for the pending restart rule.

Non-backward Compatible Changes

Manual Intervention Changes

- Renamed metric `ccp_postmaster_runtime_start_time_seconds` to `ccp_postmaster_uptime_seconds`. Both metrics report the same value, so they are currently duplicates. Note the old metric name has not yet been dropped and will still work, but it will be dropped in an upcoming version of pgMonitor.
- For PostgreSQL 9.5 & 9.6, renamed metric `ccp_wal_activity_count` to `ccp_wal_activity_total_size_bytes`. The actual value being returned has always been the total size in bytes, so the previous name was misleading. PostgreSQL 10+ already had the metric with the proper bytes size name. Note the old metric name has not yet been dropped and will still work, but it will be dropped in an upcoming version of pgMonitor.

4.2

New Features

- Add support for PostgreSQL 12
- Added new metrics (all PG versions):
  - ccp_postmaster_uptime - time in seconds since last restart of PG instance. Useful for monitoring for unexpected restarts.
  - ccp_pg_settings_checksum - monitors for changes in pg_settings
- Added new metrics (PG 9.5+ only)
  - ccp_settings_pending_restart - monitors for any settings in pg_settings in a pending_restart state
- Added new metrics (PG 10+ only)
  - ccp_pg_hba_checksum - monitors for changes in pg_hba.conf
- Added new metrics (PG 12+ only)
  - ccp_data_checksum_failure - monitors for any errors encountered for databases that have data file checksums enabled

Bug Fixes

- Use proper comparison operators in all Grafana dashboards that are using Multi-value variables.
- Change to using label_values() function on Grafana dashboard template variables. Ensures proper values in all dropdown menus are shown
- Remove changing background color of the pgBackRest panel in the PG_Details Grafana dashboard

Non-backward Compatible Changes

- New minimum required version of Grafana is now 6.5. All Grafana dashboards have been re-exported to ensure their settings are consistent and compatible with that version.

Manual Intervention Changes

- In order to use the new metrics that are available, the setup_##.sql script must be run again for your relevant version of PostgreSQL. Then all postgres_exporters services must be restarted.
- The only new rule that has been enabled by default in the Crunchy provided Prometheus rules file is `ccp_settings_pending_restart`. All other new metrics have example rules in the same file but they are commented out. Please adjust them as needed before uncommenting and using them.

4.1

- Fixed bug in PGBouncer Grafana dashboard for the Server Connection Counts Per Pool showing zero data
- Fixed Windows prometheus config file to use proper wildcard to pick up .yml files.
- Renamed Prometheus target example file to include yml extension to better ensure it is not missed. ReplicaOS.example to ReplicaOS.yml.example
- Fixed documentation to display pictures properly.

4.0

New Features

- Add pgbouncer monitoring support
  - Requires new `pgbouncer_fdw` extension provided by Crunchy Data: https://github.com/CrunchyData/pgbouncer_fdw
  - New query file can be included in QUERY_FILE_LIST: queries_pgbouncer.yml
  - New Grafana dashboard: PGBouncer.json
- Minimum version of postgres_exporter required is now 0.5.1
  - Allows connecting to multiple databases from a single exporter, however only one query file can be set per exporter service
  - If statistics are needed for per-database metrics on more than one database, recommend running a second exporter (example included as `sysconfig.postgres_exporter_pg##_per_db`) that connects to all dbs where such stats are required using separate custom query file. Leave the main exporter service to only collect global metrics from one database (preferably `postgres`).
  - DO NOT yet recommend using new `--auto-database-discovery` feature. Currently tries to connect to template databases which is never recommended.
- Added backup sizes to pgBackRest metrics that are collected by default
  - Updated pgBackRest grafana dashboard to include size graphs. Also added per-stanza dropdown filter to the top of dashboard for better readability when there are many backups.
- Added new metric to check what version of PostgreSQL the exporter is currently running on (`ccp_postgresql_version_current`).

Non-backward Compatible Changes

- Version 0.5x of postgres_exporter adds a new "server" label to all custom query output metrics. This breaks several single panel graphs that pgmonitor uses in Grafana (PG Overview, PGBackrest).

  - If upgrading, the update for the prometheus extras package must be done before upgrading to the new version of postgres_exporter. Otherwise the "server" label can cause duplication of some metrics.
  - Added a metric_relabel_configs line to the crunchy-prometheus.yml file to filter out this new label. If you are upgrading, you may have to manually add this to your own prometheus config. The package update will only automatically add this if you haven't changed the default file. Otherwise the new settings will be contained in a crunchy-prometheus.yml.rpmnew file in the package install location.

Manual Intervention Changes

- See Non-backward Compatible Changes section for update that may need to be done to prometheus config.
- Changed default DATA_SOURCE_NAME value for postgres_exporter to use the local socket for the ccp_monitoring role. This should allow the exporter to work using peer authentication, which is the default authentication method allowed by most rpm/deb provided postgres packages. This should not change any existing installations, but may affect new deployments due to new default behavior.
- Split Prometheus crunchy-alert-rules.yml file into separate node & postgres alert files to allow for more flexible rule management.
  - By default alert rules files are now looked for in `/etc/prometheus/alert-rules.d/`. Any alert files located in this folder upon restart/reload will then be picked up automatically.
  - Renamed alert files in repository to have additional .example file extension.
  - IMPORTANT UPGRADE NOTE: If upgrading with packages, prometheus may change and point to the new rules location causing your active alerts to change. Your custom alert rules have not been lost, just ensure your desired rules file(s) are moved to the new location for future compatibility.
  - Changed metric name `ccp_backrest_last_runtime` to `ccp_backrest_last_info` to reflect that it is no longer only collecting runtime stats. Note that due to metric name change, you will appear to have lost runtime history in the new grafana dashboard. The data is still there under the old metric name and can be added back as an additional data point if needed.
  - Fixed prometheus disk sizing rules to properly include ext filesystems (ext[234]). The correct syntax for the sizing-based rules is contained in the example rule files that the package provides. You will need to copy them to your current rule files if applicable.

Bug Fixes

- Disable pg_settings values that are exported by default with postgres_exporter. Fixes issue with multi-dsn support in 0.5.1 of postgres_exporter. If settings are desired as output from exporter, it is recommended to add a custom query.
- Fixed postgres_exporter service file to better parse out the destination query file name (exporter/postgres/crunchy-postgres-exporter@.service or exporter/postgres/crunchy-postgres-exporter-pg##-el6.service). Previously if any additional options were added to the OPT variable in the sysconfig, the service could throw errors on start. If you've customized your service file, please make note of changes for future compatibility.
- Update Grafana Overview dashboard to be compatible with Grafana 6.4+

3.2

- Fixed postgres_exporter service in EL6 (Redhat/CentOS) to properly use the backrest throttle environment variable in sysconfig (Github Issue #107).

3.1

- Fix broken links in Grafana OS & PG Overview Dashboards
- Updated UPGRADE steps in 3.0 release notes for new exporter service name setup. Need to re-enable service with new name and manually remove old symlink files.
- Update documentation for exporter setup to use new service names

3.0

- New minimum version requirements for software that is part of pgmonitor are as follows, including links to release notes:
  - Prometheus: 2.9.2 - https://github.com/prometheus/prometheus/releases
  - Alertmanager: 0.17.0 - https://github.com/prometheus/alertmanager/releases
  - Grafana: 6.1.6 (major version change from 5.x) - https://community.grafana.com/t/release-notes-v6-1-x/15772
  - node_exporter: 0.18.0 - https://github.com/prometheus/node_exporter (Note breaking changes for some metrics. None of those broken are used by default in pgmonitor).
- The service file for postgres_exporter provided by pgmonitor has been renamed to make it more consistent with typical systemd service names.
  - IMPORTANT: See upgrade notes below about changes to sysconfig file before restarting service!
  - Only applies to systemd file for RHEL/CentOS 7

  - Changed crunchy_postgres_exporter@.service to crunchy-postgres-exporter@.service (underscores to dashes).
  - Note that you will need to use the new service name to interact with it from now on. This requires enabling the new service name and restarting it:
    * `systemctl enable crunchy-postgres-exporter@postgres_exporter_pg11`
    * `systemctl restart crunchy-postgres-exporter@postgres_exporter_pg11`
  - Due to the removal of the old service file, you cannot use systemctl to disable the old service. Instead just remove the symlinks manually:
    * 'rm /etc/systemd/system/multi-user.target.wants/crunchy_postgres_exporter@*
- The single query.yml file used by postgres_exporter to use Crunchy's custom queries is now dynamically generated automatically upon service start/restart.
  - A new variable, QUERY_FILE_LIST, is now set in the sysconfig file for the service. It is a space delimited list of the full paths to all query files that will be concatenated together. See sysconfig file for several examples and a recommended default to set.
  - This now ensures that any updates to desired query files will be automatically applied when the package is updated and the service is restarted without having to manually rebuild the query.yml file.
  - This new variable is not required and you can continue to manually manage your queries.yml file. Ensure that the QUERY_FILE_LIST variable is not set if this is desired.
  - UPGRADE NOTES:
    * Backup your current queries.yml file.
    * If you have not modified the default sysconfig file for your postgres_exporter service (/etc/sysconfig/postgres_exporter_pg##) updating to 3.0 will overwrite your current sysconfig file and put the default QUERY_FILE_LIST value in place, possibly overwriting your current queries.yml file. Again, please ensure you backup your current queries.yml file and then set the QUERY_FILE_LIST variable appropriately to dynamically generate your queries file for you in the future. Or unset the variable and continue managing it manually.
    * If you have modified your sysconfig file from what the package provides, it will not be overwritten and a new sysconfig file with an `.rpmnew` extension will be created. You can reference this .rpmnew file for how to update your sysconfig file to take advantage of the new QUERY_FILE_LIST option.
    * Ensure all postgres_exporters you have running set the QUERY_FILE_LIST properly if using it. Especially if multiple exporters are using the same query file.
- Prometheus targets for pgmonitor provided exporters (postgres_exporter & node_exporter) have had labels added to them for use in pgmonitor provided Grafana Dashboards.
  - Added new label `exp_type` (export type) in prometheus targets to better distinguish OS and Postgres metrics in Prometheus. Possible current values are `pg` or `node`.
  - UPGRADE NOTES: This new label must be applied to your Prometheus target files if you are using the Grafana dashboards provided by pgmonitor. Note that if you previously defined node and postgres_exporter targets under a single target, you will now need to separate them, keeping the same job name for both. See example target files provided in package/repo for how to apply new label (Ex. ProductionDB.yml.example & ProductionOS.yml.example).
  - If you are not using the pgmonitor provided Grafana dashboards, these new labels are optional.
- Grafana Dashboards Updates
  - New dashboards require at least Grafana 6.x.
  - UPGRADE NOTES: Once new Prometheus label (mentioned above) is applied, dashboard provisioning should take care of updating all dashboards once the new ones are in place. Note that all dashboards provided by pgmonitor 3.0+ now assume this new label and will not work until the Prometheus exp_type label is added.
  - Renamed dashboard files for better naming consistency. Dashboard titles also updated accordingly.
    * UPGRADE NOTES: If installing from package, it will take care of care of renaming dashboard files. Otherwise, dashboards have been renamed as follows below. Ensure old files are renamed/removed to avoid duplicating/breaking current dashboards. Easiest manual update method is to remove all dashboards provided by pgmonitor and copy all new ones back. Provisioning will then take care of updating things for you.
    * renamed: BloatDetails.json -> Bloat_Details.json
    * renamed: FilesystemDetails.json -> Filesystem_Details.json
    * renamed: PostgreSQLDetails.json -> PG_Details.json
    * renamed: PostgreSQL.json -> PG_Overview.json
    * renamed: TableSize_Detail.json -> TableSize_Details.json
  - Dashboard names have been updated to match with new naming consistency. If you had direct links to dashboards, these may need to be updated.
  - Split OS Metrics into their own dashboard separate from PG Metrics.
  - Added link to PGbackrest dashboard to top of Postgres Details Dashboard. Link shows time since last successful backup (any type) for that target system.
  - Added new OS Details dashboard
  - Added new etcd dashboard
  - Add new Top Level Overview dashboard that links to all other Overview dashboards
  - Set default refresh rate for most dashboards to 15 minutes.
  - Obsolete "jobname" grafana variable in all dashboards. Add new grafana variables pgnodes, osnodes that use the new labels added in prometheus targets notded above.
- New configuration option for postgres_exporter sysconfig file to control PGBackrest refresh rate

  - PGBACKREST_INFO_THROTTLE_MINUTES
  - This is the value, in minutes, passed along to the monitor.pgbackrest_info() function in all backrest checks
  - Default is 10 minutes

2.4

- UPGRADE NOTE: All exporter issues below can be fixed by re-running the setup_pg##.sql file for your major version of postgres. For the pgbackrest fix, you will also need to update the queries.yml file for the exporter to include the new queries found in the queries_backrest.yml file.
- Fixed several issues with pgbackrest monitor in postgres_exporter that was included in pgmonitor v2.3
  - Fixed incorrect data being returned by monitor query on PostgreSQL 9.6 and earlier. The same, latest backup time was being returned for all stanzas instead of returning the time per stanza.
  - Fixed backrest query causing the postgres_exporter to hang and cause all metric output to stop.
  - Fixed backrest monitor to work with larger amount of data being returned by the "pgbackrest info" command. Previously, once returned data size reached a certain point, would cause a "missing chunk" error.
  - Added a parameter to the function that is called to control how often the underlying info command is actually run. On systems with high backup counts, info can be a slightly more expensive call. This helps to control that, no matter what the scrape interval of prometheus is set to. Default is to get new data every 10 minutes, otherwise just queries from an internal table that stores the last info run.
  - Backrest monitoring can now be run on replicas as well, but cannot update the current backup status since that requires writing to the database. This is mostly to enable monitoring setups to be consistent between primary/replica in case of failover.
- Fixed issue with ccp_sequence_exhaustion metric that would cause postgres_exporter output to hang if any table that contained a sequence was dropped during a long running transaction.
- Added new metric (ccp_replication_slots) and alert (PGReplicationSlotsInactive) for monitoring replication slot status. New metric and alert can be found in queries_pg##.yml and crunchy-alert-rules.yml respectively.
- Added lock_timeout of 2 minutes to the ccp_monitoring role. Avoids monitoring causing any extensive lock interference with normal database operations.
- Added Grafana Dashboard for PGBackrest status information.
- Fixed lines being hidden in the "Total Bloat %" graph in BloatDetails Grafana dashboard.
- Removed unnecessary drilldown link in Total Bloat % graph in BloatDetails Grafana dashboard.

2.3

- Fixed bug in Prometheus alerts that was causing some of them to be stuck in PENDING mode indefinitely and never firing. This unfortunately removes the current alert value from the Grafana Prometheus Alerts dashboard.
  - If you can't simply overwrite your current alerts configuration file with the one provided, remove the following option from every alert: `alert_value: '{{ $value }}'`
- Added feature to monitor pgbackrest backups (https://pgbackrest.org)
  - Separate metrics exist to monitor for the latest full, incremental and/or differential backups. Note that a full will always count as both an incremnetal and diff and a diff will always count as an incremental.
  - Another metric can monitor the runtime of the latest backup of each type per stanza.
  - Run the setup_pg##.sql file again in the database that your exporter(s) connect to to install the new, required function: "monitor.pgbackrest_info()". It has security definer so execution privileges can be granted as needed, but it must be owned by a superuser.
  - New metrics are located in the exporter/postgres/queries_backrest.yml file. Add the one(s) you want to the main queries file being used by your currently running exporter(s) and restart.
  - Example alert rules for different backup scenarios have been added to the prometheus/crunchy-alert-rules.yml file. They are commented out to avoid false alarms until valid backup settings for your environment are in place.
- Added new feature to monitor for failing archive_command calls.
  - New metric "ccp_archive_command_status" is located in exporter/postgres/queries_common.yml. Add this to the main queries file being used by your currently running exporter(s) and restart.
  - A new alert rule "PGArchiveCommandStatus" has been added to the prometheus/crunchy-alert-rules.yml file.
- Added new feature to monitor for sequence exhaustion
  - Requires installation of a new function located in the setup_pg##.yml file for your relevant major version of PostgreSQL. Must be installed by a superuser.
  - New metric "ccp_sequence_exhaustion" located in exporter/postgres/queries_common.yml. Add this to the main queries file being used by your currently running exporter(s) and restart. A new alert rule "PGSequenceExhaustion" has been added to the prometheus/crunchy-alert-rules.yml file.
- The setup_pg##.sql file now has logic to avoid throwing errors when the ccp_monitoring role already exists. Also always attempts to drop the functions it manages first to account for when the function signature changes in ways that OR REPLACE doesn't handle. All this allows easier re-running of the script when new features are added or used in automation systems. Thanks to Jason O'Donnell for role logic.

2.2

- Fixed broken ccp_wal_activity check for PostgreSQL 9.4 & 9.5. Updated check is located in the relevant exporter/postgres/-queries_pg##.yml file
- Fixed broken service files for postgres_exporter on RHEL6 systems.
- Removed explicit "public" schema in ccp_bloat_check query so that it will properly use the search_path in case bloat tables were installed in another schema
- Removed query files for PostgreSQL versions no longer supported by pgmonitor (9.2 & 9.3)

2.1

- **IMPORTANT UPGRADE NOTE FOR CRUNCHY PACKAGE USERS**: In version 2.0, the Crunchy provided extras for node_exporter were split out from the pgmonitor-pg##-extras package. A dependency was kept between these packages to make upgrading easier. For 2.1, the dependency between these packages has been removed. When upgrading from 1.7 or earlier, if you have node_exporter and postgres_exporter running on the same systems, ensure that you install the separate pgmonitor-node_exporters_extras package after the update. See the README for the full package name(s).
- Minimum required versions of software used in pgmonitor have been updated to:
  - Prometheus 2.5.0
  - Prometheus Alertmanager 0.15.3
  - postgres_exporter 0.4.7 (enables full PG11 support)
  - Grafana 5.3.4.
- Fixed Grafana data source to use the "proxy" mode instead of "direct" with default install. Should fix connection issues encountered during default setup between Grafana & Prometheus.
- Renamed functions_pg##.sql file to setup_pg##.sql to better clarify what it's for (and because it's not just functions).
- Added ccp_wal_activity metric to help monitor WAL generation rate.
  - For all PG versions, provides total current size of WAL directory. For PG10+, it also provides the size of WAL generated in the last 5 minutes
  - Note that for PG96 and lower, a new security definer function must be added (can just run setup_pg##.sql again).
  - New metric definition is located in the queries_pg##.yml file.
  - No default rules have been added since this is very use-case dependent.
- Improved accuracy of "Idle In Transaction" monitoring times in PostgreSQL. Base the time measured on the state change of the session vs the total transaction runtime.
- Split setup_pg92-96.sql and queries_pg92-96.sql into individual files per major version.
- Added commented out example prometheus alert rule for checking if a postgres database has changed from replica to primary or vice versa. Must be set on a per system basis since you have to tell it if a system is supposed to be a primary or replica.
- Removed pg_stat_statements prometheus metric and security definer function from setup script. We highly recommend having pg_stat_statements installed on a database, and we still include its installation in the documentation, but we currently don't have any useful metric recommendations from it to collect in prometheus.
- Added some default filters for the bloat check cronjob to avoid unnecessary waste in the prometheus storage of bloat metrics.
- Update documentation.

2.0

- Recommended version of Prometheus is now 2.3.2. Recommended version of Alertmanager is 0.15.1. Recommended version of postgres_exporter is 0.4.6.
- Upgrade required version of node_exporter to minimum of 0.16.0. Note that many of the metrics that are used in Grafana and Prometheus alerting have had their names changed.
  - This version adds these new metrics into Grafana graphs without removing the old metric names on most, but not all, graphs. This allows trending history to be kept. Note that line colors will change in graphs and legend names will be duplicated until the old metric data is expired out.
  - Prometheus alerts have been set to use the new metric names since the alerts are based only on recent values.
  - IMPORTANT: A future pgmonitor update will remove these old metric names from Grafana graphs, so please ensure these changes are accounted for in your architecture.
  - See full release notes for 0.16.0 - https://github.com/prometheus/node_exporter/releases/tag/v0.16.0
- The postgres_exporter service no longer uses a symlink in /etc/sysconfig to point to a default "postgres_exporter" file. This was causing issues with several upgrade scenarios. New installation instructions now have the service pointing directly to the relevant sysconfig file for the major PostgreSQL version.
  - **IMPORTANT**: If you are using the default postgres_exporter service, you will need to update your service name so it uses the proper sysconfig file. See the README file for the new default service name in the "Enable Services" section and run

the “enable” command found there. You should then also disable/remove the old service so it doesn’t try to start again in the future.

- The additional Crunchy provided configurations for node_exporter have been split out from the pgmonitor-pg##-extras package to the pgmonitor-node_exporter-extras package. This was done to allow multiple versions of the pg##-extras package to be installed with different major versions of Postgres. There is still currently a dependency that the node extras packages must be installed with the pg##-extras so that upgrading doesn’t break existing systems. This dependency will be revisited in the future.
- Removed the requirement for a shell script to monitor if the database is up and its status as either a primary or replica. Up status is now using the native “pg_up” metric from postgres_exporter and a new metric query was written for checking the recovery status of a system (ccp_is_in_recovery).
  - The PostgreSQL.json overview dashboard that used this metric has been redesigned. Unfortunately it can no longer be colored RED for down systems, only go colorless and say “DOWN”. This is a known limitation of handling null metric values in Grafana and part of a larger fix coming in future versions - https://github.com/grafana/grafana/issues/11418
- Upgrade required version of Grafana to minimum of 5.2.1.
  - All provided dashboards require this minimum version to work.
  - If you notice that links between the dashboards are broken after the upgrade, clear your browser’s cache. The 301 redirects used between dashboards can get cached and they have changed in the new major version.
  - See extensive release notes for major version changes in Grafana - https://community.grafana.com/t/release-notes-v-5-1-x
- Change Grafana datasource and dashboard installation to use provisioning vs manual setup via the web interface. Note this means that future updates to the provided datasources and dashboards must be done through config files as well. Or they can be saved as a new dashboard for more extensive customization.
- Change recommended configuration for Grafana to use PostgreSQL as database backend. Updated installation documentation.
- Added Prometheus Alerts Dashboard. Shows both active alerts and 1 week history in table format.
- Removed Gauges from PostgreSQLDetails Dashboard. “Current” value was not being shown properly and gauges were misleading in their values depending on the time range chosen. For a quick glance to see if there are any problems, be sure to set your alert thresholds properly and use the new Prometheus Alerts Dashboard.
- Added max_query_time metric to track long running queries in general. Also added an alert for that metric to crunchy prometheus alerts.
- Added “IO Time Per Device in Seconds” graph to Filesystems dashboard.
- Fixed Memory and Swap Graphs on PostgreSQLDetails dashboard to more accurately show used resources. History for these graphs before this upgrade is not being shown since it is no longer graphing the same data.
- Crontabs are no longer PostgreSQL major version dependent at this time. Consolidated down to a single crontab file for all versions.
- Removed unnecessary functions from functions_pg10.sql. All queries in queries_pg10.yml currently only require the pg_monitor system role to be granted and have been updated with this assumption.
- Changed default cron runtime of pg_bloat_check to once a week on early morning weekend.
- Change PostgreSQL overview dashboard to use background colors instead of gauges for better visibility.
- Fixed permission issues with /etc/postgres_exporter folder to allow ccp_monitoring system user better control.

1.7

- Fixed duplicate and incorrect replication byte lag queries. The one contained in queries_common.yml should not have been there. It should be in queries_pg92-96.yml, but there was also one already there. However, the one already in pg92-96 was incorrect since prior to PG10, it requires superuser/security definer to fully access replication statistics. Corrected the version specific file to have the correct query. Made the query in the pg10 file consistent. Ensure you update your generated queries.yml file with he new queries.
- Fixed the PostgreSQLDetails.json dashboard to use the correct replication byte lag metric (referencing above fix). The easiest way to fix this is to delete this dashboard and re-import it. Otherwise, if you’ve made customizations you don’t want to lose, you can grab the correct metric query from the updated dashboard gauge and edit your existing dashboard to use it.
- The combination of the above two fixes corrects the pgmonitor setup being able to properly handle there being multiple replicas from a single primary. Previously this would cause postgres_exporter to throw duplicate metric errors.
- Fixed the query in queries_bloat.yml to be able to properly handle if there was a bloat amount larger than max int4 bytes. Ensure you update your generated queries.yml file with the new query.

1.6

- Fixed formatting bug in crunchy-prometheus.yml. Thanks to Doug Hunley for reporting the issue.

1.5

- Add support for disabling built in queries in postgres_exporter 0.4.5. Also explicitly ignore these metrics via a prometheus filter so they’re not ingested even if new option isn’t used. This means that v1.5 of pgmonitor now requires 0.4.5 of postgres_exporter by default.
- Improved exporter down alert to avoid unnecessary alerts for brief outages that resolve themselves quickly.
- Added new FilesystemDetails dashboard for grafana that is linked to from the Filesystem graph on PostgreSQLDetails.
- Top level PostgreSQL grafana dashboard now identifies whether a system is read/write or readonly to better distinguish primary/replica systems.
- Added instructions for non-packaged installation using pgmonitor configuration files.

- Revised and better formatted README documentation

1.4

- Fixed filesystem graphs in PostgreSQLDetails dashboard
- Cosmetic changes to PostgreSQLDetails dashboard
- Added instructions for importing dashboards via Grafana API

1.3

- Fixed error in PG10 queries file.
- Fixed disk usage alert for prometheus to work better when there are many jobs with similar mountpoints. Also fixed syntax error in warning alert.
- Moved connection stats query from common to version specific queries due to PG10 differences. Clarified naming of files for which versions they work for.
- Added dropdown for the Job to the lower level drill down dashboards in Grafana. Allows selecting of a specific system from the dashboard itself without having to click through on a higher level.
- Removed pg_stat_statements graph from PostgreSQLDetails dashboard. Needs refinement to make it more useful.

1.2

- Change service and sysconfig files to use single OPT environment variable instead of one variable per cmd option
- Fix error in PG10 monitoring functions file
- Initial version of Prometheus 2.0 job deletion script. Requires API call not available yet in 2.0.0 for full functionality

1.1

- Implement rpmnew/rpmsave feature instead of using .example files to prevent package overwriting user changes to configs

1.0

- Initial stable release