

Contents

pgmonitor	2
pgmonitor is your all-in-one tool to easily create an environment to visualize the health and performance of your PostgreSQL cluster.	2
Contents	3
Purpose	4
Supported Platforms	4
Operating Systems	4
PostgreSQL	4
Installation	4
1. Prometheus	5
2. exporter	5
3. Grafana	5
Roadmap	5
Version History	5
Sponsors	5
Legal Notices	5
Setting up exporters for pgmonitor	6
Installation	6
Upgrading	6
Installation on RHEL/CentOS 7	7
Setup	9
Setup on RHEL/CentOS 7	9
Running multiple postgres exporters (RHEL / CentOS 7)	12
Note for packaging (RHEL/CENTOS 7)	13
Installation / Setup on RHEL/CentOS 6	13
Running multiple postgres exporters (RHEL / CentOS 6)	14

Grafana	14
Installation	15
Upgrading	15
With RPM Packages	15
Setup	16
Configuration Database	16
Datasource & Dashboard Provisioning	17
Setting up Prometheus for pgmonitor	17
Installation	17
Upgrading	17
Installation on RHEL/CentOS 7	18
Setup	20
Setup on RHEL/CentOS 7	20
Note for packaging (RHEL/CentOS 7)	21
Setup on RHEL/CentOS 6	21

pgmonitor

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

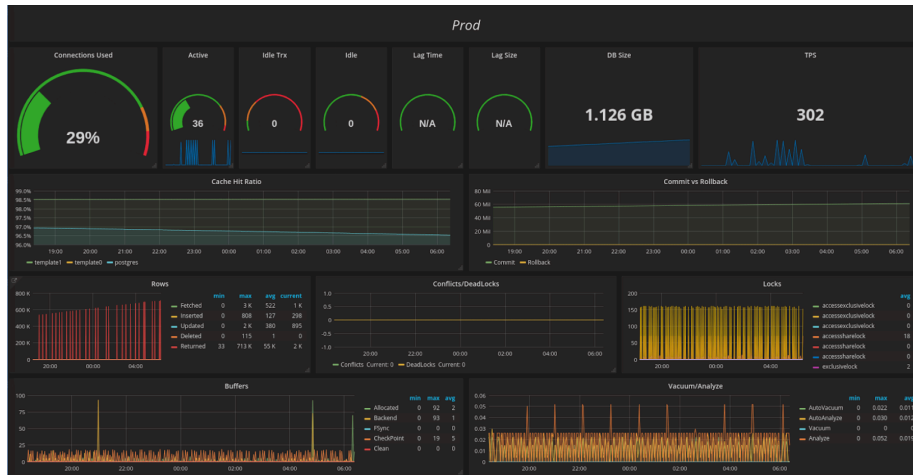


Figure 1: pgmonitor

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
 - Supported Platforms
 - Operating Systems
 - PostgreSQL
 - Installation
 - Roadmap
 - Version History
 - Sponsors
 - Legal Notices
-

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 packages and necessary configuration 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.
- [PostgreSQL Exporter](#) - an open-source data export to Prometheus that supports collecting metrics from any PostgreSQL server version 9.1 and above.

Supported Platforms

Operating Systems

- Prometheus/Alertmanager & Grafana: CentOS/RHEL 7 or greater
- Exporters (node & postgres): CentOS/RHEL 6 or greater

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.
- 11, 10, 9.6, 9.5, 9.4

Known issues

- PostgreSQL 10+ SCRAM-SHA-256 encrypted password are not yet supported by underlying go library used by postgres_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. [Prometheus](#)
2. [exporter](#)
3. [Grafana](#)

Roadmap

- Additional monitoring metrics out-of-the-box
- Improved visualizations
- Project build testing

Version History

For the [full history](#) of pgmonitor, please see the [CHANGELOG](#).

Sponsors



Figure 2: 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

Copyright © 2018 Crunchy Data Solutions, Inc.

CRUNCHY DATA SOLUTIONS, INC. PROVIDES THIS GUIDE “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES

OF NON INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Crunchy, Crunchy Data Solutions, Inc. and the Crunchy Hippo Logo are trademarks of Crunchy Data Solutions, Inc.

Setting up exporters for pgmonitor

The exporters below can be set up on any Linux-based system, but the instructions below use RHEL/CentOS 7.

- [Installation](#)
- [Setup](#)
- [RHEL / CentOS 7](#)
- [RHEL / CentOS 6](#)

Installation

Upgrading

1.x -> 2.x

- See CHANGLOG.md file for full details on what has changed in this major version upgrade.
- Many of the metric names in node_exporter v0.16.0 have had their names changed. All of the ones that pgmonitor uses in alerting and grafana related to CPU, Memory and Disk have been renamed. All files provided by pgmonitor 2.x have been updated to account for these changes so please either use these new files or see what has changed and incorporate them into your environment.
- The symlink for the postgres_exporter sysconfig file is no longer being used. The symlink is removed as part of the upgrade, so the default postgres_exporter service that previously used this may have to be updated. See the **Enable Services** section below for the correct systemctl command to create the new service name. The old service can then be disabled/removed.
- The `ccp_is_ready` check has been removed and pgmonitor now uses the `pg_up` check built into postgres_exporter. Prometheus alerting and grafana dashboards have been updated to account for this.
- A new metric `ccp_is_in_recovery` is used to help determine the primary/replica state of a given database in the grafana dashboards. The query for this can be found in `queries_common.sql`

Installation on RHEL/CentOS 7

With RPM Packages There are RPM packages available to [Crunchy Data](#) customers through the [Crunchy Customer Portal](#).

If you install the below available packages with RPM, you can continue reading at the [Setup](#) section.

Package Name	Description
node_exporter	Base package for node_exporter
postgres_exporter	Base package for postgres_exporter
pgmonitor-pg###-extras	Crunchy optimized configurations for postgres_exporter. Note that ea
pgmonitor-node_exporter-extras	Crunchy optimized configurations for node_exporter
pg_bloat_check	Package for pg_bloat_check script

Available Packages

Without Packages For non-package installations, the exporters & pg_bloat_check can be downloaded from their respective repositories:

Library	
node_exporter	https://github.com/prometheus/node_exporter/releases
postgres_exporter	https://github.com/wrouesnel/postgres_exporter/releases
pg_bloat_check	https://github.com/keithf4/pg_bloat_check

User and Configuration Directory Installation You will need to create a user named `ccp_monitoring` which you can do with the following command:

```
sudo useradd ccp_monitoring
```

Create a folder in `/var/lib/` and set its permissions as such:

```
sudo mkdir /var/lib/ccp_monitoring
sudo chmod 0700 /var/lib/ccp_monitoring
sudo chown ccp_monitoring /var/lib/ccp_monitoring
```

Configuration File Installation All executables are expected to be in the `/usr/bin` directory. A base `node_exporter` systemd file is expected to be in place already. An example one can be found here:

`https://github.com/lest/prometheus-rpm/tree/master/node_exporter`

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: 95, 96, 10, etc.).

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

```
sudo mkdir /var/lib/ccp_monitoring/node_exporter
sudo chmod 0700 /var/lib/ccp_monitoring/node_exporter
sudo chown ccp_monitoring /var/lib/ccp_monitoring/node_exporter
```

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

pgmonitor Configuration File	System Location
<code>node/crunchy-node-exporter-service-el7.conf</code>	<code>/etc/systemd/system/node_exporter.service.d/crunchy</code>
<code>node/sysconfig.node_exporter</code>	<code>/etc/sysconfig/node_exporter</code>

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

pgmonitor Configuration File	System Location
<code>crontab.txt</code>	<code>/etc/postgres_exporter/##/crontab.txt</code>
<code>postgres/crunchy_postgres_exporter@.service</code>	<code>/usr/lib/systemd/system/crunchy_postgres_exporter@.service</code>
<code>postgres/sysconfig.postgres_exporter_pg##</code>	<code>/etc/sysconfig/postgres_exporter_pg##</code>
<code>postgres/setup_pg###.sql</code>	<code>/etc/postgres_exporter/##/setup_pg###.sql</code>
<code>postgres/queries_pg###.yaml</code>	<code>/etc/postgres_exporter/##/queries_pg###.yaml</code>
<code>postgres/queries_common.yaml</code>	<code>/etc/postgres_exporter/##/queries_common.yaml</code>
<code>postgres/queries_per_db.yaml</code>	<code>/etc/postgres_exporter/##/queries_per_db.yaml</code>
<code>postgres/queries_bloat.yaml</code>	<code>/etc/postgres_exporter/##/queries_bloat.yaml</code>

Setup

Setup on RHEL/CentOS 7

Service Configuration The following files contain defaults that should enable the exporters 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: `- /etc/systemd/system/node_exporter.service.d/crunchy-node-exporter-service-environment.d/`
`- /etc/sysconfig/node_exporter` - `/etc/sysconfig/postgres_exporter_pg##`

Note that `/etc/sysconfig/postgres_exporter_pg##` is the default sysconfig file for monitoring the database running on the default port 5432 and connects to the “postgres” database. If you’ve installed the pgmonitor setup to a different database, modify this file accordingly or make a new one. If you make a new one, ensure the service name you enable references this file (see the Enable Services section below).

Database Configuration

General Configuration First, make sure you have installed the PostgreSQL contrib modules. You can install them with the following command:

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

Where `##` corresponds to your current PostgreSQL version. For PostgreSQL 10 this would be:

```
sudo yum install postgresql10-contrib
```

You will need to modify your `postgresql.conf` configuration file to tell PostgreSQL to load shared libraries. In the default setup, this file can be found at `/var/lib/pgsql/10/data/postgresql.conf`.

Modify your `postgresql.conf` configuration file to add the following shared libraries

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

You will need to restart your PostgreSQL instance for the change to take effect. Neither of the above extensions are used outside of the postgres database itself, but we find they are extremely useful to have loaded and available in the database when further diagnosis of issues is required.

For each database you are planning to monitor, you will need to run the following command as a PostgreSQL superuser:

```
CREATE EXTENSION pg_stat_statements;
```

If you want the `pg_stat_statements` extension to be available in all newly created databases, you can run the following command as a PostgreSQL superuser:

```
psql -d template1 -c "CREATE EXTENSION pg_stat_statements;"
```

Query File	Description
setup_pg###.sql	Creates <code>ccp_monitoring</code> role with all necessary grants. Creates any extra monitoring roles.
queries_bloat.yml	postgres_exporter query file to allow bloat monitoring.
queries_common.yml	postgres_exporter query file with minimal recommended queries that are common to all versions.
queries_per_db.yml	postgres_exporter query file with queries that gather per database stats. WARNING: This file is not recommended for production use.
queries_pg###.yml	postgres_exporter query file for queries that are specific to the given version of PostgreSQL.

Monitoring Setup Install the `setup_pg###.sql` script to all databases you will be monitoring in the cluster. The queries common to all postgres versions are contained in `queries_common.yml`. Major version specific queries are contained in a relevantly named file. Queries for more specialized monitoring are contained in additional files. `postgres_exporter` only takes a single query file as an argument for custom queries, so cat together the queries necessary into a single file.

For example, to use just the common queries for PostgreSQL 9.6 do the following. Note the location of the final queries file is based on the major version installed. The exporter service will look in the relevant version folder in the `/etc/postgres_exporter` directory:

```
cd /etc/postgres_exporter/96
cat queries_common.yml queries_per_db.yml queries_pg96.yml > queries.yml
psql -f /etc/postgres_exporter/96/setup_pg96.sql
```

As another example, to include queries for PostgreSQL 10 as well as bloat do the following:

```
cd /etc/postgres_exporter/10
cat queries_common.yml queries_per_db.yml queries_pg10.yml queries_bloat.yml > queries.yml
psql -f /etc/postgres_exporter/10/setup_pg10.sql
```

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

Access Control: GRANT statements

The `ccp_monitoring` database role (created by running the “`setup_pg###.sql`” file above) must be allowed to connect to all databases in the cluster. To do this, 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;
```

Bloat setup

Run script on the specific database(s) you will be monitoring bloat for in the cluster. See special note in `crontab.txt` concerning a superuser requirement 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_st
psql -d postgres -c "GRANT SELECT,INSERT,UPDATE,DELETE,TRUNCATE ON bloat_indexes, bloat_
```

The `/etc/postgres_exporter/##/crontab.txt` file is meant to be a guide for how you setup the `ccp_monitoring` *crontab*. You should modify *crontab* entries to schedule your bloat check for off-peak hours. This script is meant to be run at most, once a week. Once a month is usually good enough for most databases as long as the results are acted upon quickly.

The script requires being run by a database superuser by default since it must be able to run a scan on every table. If you'd like to not run it as a superuser, you will have to create a new role that has read permissions on all tables in all schemas that are to be monitored for bloat. You can then change the user in the connection string option to the script.

Enable Services

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

To most easily allow the possibility 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 EnvironmentFile is named “postgres_exporter_pg##” and tied to the major version of postgres that it was installed for. Be sure to replace the ## in the below commands first!

```
sudo systemctl enable crunchy_postgres_exporter@postgres_exporter_pg##.service
sudo systemctl start crunchy_postgres_exporter@postgres_exporter_pg##
sudo systemctl status crunchy_postgres_exporter@postgres_exporter_pg##
```

Running multiple postgres exporters (RHEL / CentOS 7)

Certain metrics are not cluster-wide, so in that case multiple exporters must be run to collect all relevant metrics. The queries_per_db.yml file contains these queries and the secondary exporter(s) can use this file to collect those metrics and avoid duplicating cluster-wide metrics. Note that some other metrics are per database as well (bloat). You can then define multiple targets for that job in Prometheus so that all the metrics are collected together. Note that the “setup_*.sql” file does not need to be run on these additional databases.

```
cd /etc/postgres_exporter/96
cat queries_per_db.yml queries_bloat.yml > queries_mydb.yml
```

You’ll need to create a new sysconfig environment file for the second exporter service. You can just copy the existing ones and modify the relevant lines, mainly being the port, database name, and query file

```
cp /etc/sysconfig/postgres_exporter_pg## /etc/sysconfig/postgres_exporter_mydb
```

```
OPT="--web.listen-address=0.0.0.0:9188 --extend.query-path=/etc/postgres_exporter/96/queries_mydb.yml"
DATA_SOURCE_NAME="postgresql://ccp_monitoring@localhost:5432/mydb?sslmode=disable"
```

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_mydb.service
sudo systemctl start crunchy_postgres_exporter@postgres_exporter_mydb
sudo systemctl status crunchy_postgres_exporter@postgres_exporter_mydb
```

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

Note for packaging (RHEL/CENTOS 7)

The service override file(s) must be placed in the relevant drop-in folder to override the default service files.

```
/etc/systemd/system/node_exporter.service.d/*.conf
```

After a daemon-reload, systemd should automatically find these files and the crunchy services should work as intended.

Installation / Setup on RHEL/CentOS 6

The `node_exporter` and `postgres_exporter` services on RHEL6 require the “daemonize” package that is part of the EPEL repository. This can be turned on by running:

```
sudo yum install epel-release
```

All setup for the exporters is the same on RHEL6 as it was for 7 with the exception of the base service files. Whereas RHEL7 uses systemd, RHEL6 uses `init.d`. The Crunchy RHEL6 packages will create the base service files for you

```
/etc/init.d/crunchy-node-exporter /etc/init.d/crunchy-postgres-exporter
```

Note that these service files are managed by the package and any changes you make to them could be overwritten by future updates. If you need to customize the service files for RHEL6, it’s recommended making a copy and editing/using those.

Or if you are setting this up manually, the repository file locations and expected directories are:

```
node/crunchy-node-exporter-el6.service -> /etc/init.d/crunchy-postgres-exporter
postgres/crunchy-postgres-exporter-el6.service -> /etc/init.d/crunchy-postgres-exporter

/var/run/postgres_exporter/
/var/log/postgres_exporter/ (owned by postgres_exporter service user)

/var/run/node_exporter/
/var/log/node_exporter/ (owned by node_exporter service user)
```

The same `/etc/sysconfig` files that are used in RHEL7 above are also used in RHEL6, so follow guidance above concerning them and the notes that are contained in the files themselves.

Once the files are in place, set the service to start on boot, then manually start it

```
sudo chkconfig crunchy-node-exporter on
sudo service crunchy-node-exporter start
sudo service crunchy-node-exporter status

sudo chkconfig crunchy-postgres-exporter on
sudo service crunchy-postgres-exporter start
sudo service crunchy-postgres-exporter status
```

Running multiple postgres exporters (RHEL / CentOS 6)

If you need to run multiple postgres_exporter services, follow the same instructions as RHEL / CentOS 7 for making a new queries_XX.yml file to only gather database specific metrics. Then follow the steps below:

- Make a copy of the /etc/sysconfig file with a new name
- Update --web.listen-address in the new sysconfig file to use a new port number
- Update --extend.query-path in the new sysconfig file to point to the new query file generated
- Update the DATA_SOURCE_NAME in the new sysconfig file to point to the name of the database to be monitored
- Make a copy of the /etc/init.d/crunchy-postgres-exporter with a new name
- Update the SYSCONFIG variable in the new init.d file to match the new sysconfig file
- Update the Prometheus auto.d target file to include the new exporter in the same one you already had running for this system

Remaining steps to initialize service at boot and start it up should be the same as above for the default service.

Grafana

There are RPM packages available to [Crunchy Data](#) customers through the [Crunchy Customer Portal](#). Otherwise the Grafana RPM Package can be downloaded and installed from <https://grafana.com/grafana/download>. There is no difference between the Crunchy provided package and the one directly from Grafana.

Installation

Upgrading

If you'd like to take advantage of the new provisioning system in Grafana 5 provided by pgmonitor 2.x, we recommend either renaming or deleting your existing datasources and dashboards so there are no issues when the provisioned versions are imported.

When upgrading from pgmonitor 1.x to 2.x, note that many of the system related metric names from node_exporter have had their names changed. The new graphs provided for Grafana 5+ have taken these new names into account. Also, the top level PostgreSQL Overview dashboard no longer uses the ccp_is_ready metric, so you will have to include some new postgres_exporter metrics for that dashboard to work.

With RPM Packages

There are RPM packages available to [Crunchy Data](#) customers through the [Crunchy Customer Portal](#).

If you install the below available packages with RPM, you can continue reading at the [Setup](#) section.

Package Name	Description
grafana	Base package for grafana
pgmonitor-grafana-extras	Crunchy configurations for datasource & dashboard provisioning

Available Packages

Without Packages Create the following directories on your grafana server if they don't exist:

```
mkdir -p /etc/grafana/provisioning/datasources mkdir -p /etc/grafana/provisioning/dashboards
mkdir -p /etc/grafana/crunchy_dashboards
```

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

Review the `crunchy_grafana_datasource.yml` file to ensure it is looking at your Prometheus database. The included file assumes Grafana and Prometheus are running on the same system. DO NOT CHANGE the datasource “name” if you will be using the dashboards provided in this repo. They assume that name and will not work otherwise. Any other options can be changed as needed. Save the `crunchy_grafana_datasource.yml` file and rename to `/etc/grafana/provisioning/datasources/datasources.yml`. Restart grafana and confirm through the web interface that the datasource was provisioned and working.

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

Save all of the .json dashboard files to the `/etc/grafana/crunchy_dashboards` folder.

Setup

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 5.x 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 can no longer 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 exporting and importing manually via the web interface.

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. If you're upgrading from Grafana 4.x to 5.x, you will have to add the “provisioning” option to the `[paths]` section of the `grafana.ini` file. Once that is done, just restart grafana and all datasources and dashboards should appear.

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

Setting up Prometheus for pgmonitor

Prometheus can be set up on any Linux-based system, but the instructions below use RHEL/CentOS 7.

- [Installation](#)
- [Setup](#)
- [RHEL / CentOS 7](#)

Installation

Upgrading

When upgrading from pgmonitor 1.x to 2.x, note that the alerting rules for `node_exporter` metrics have had many of their names changed. If you've changed the provided alerting rules file, installing the new package should

create a file called `/etc/prometheus/crunchy-alert-rules.yml.rpmnew` and not overwrite your current file. You should be able to copy the new rules as needed from there.

Installation on RHEL/CentOS 7

With RPM Packages There are RPM packages available to [Crunchy Data](#) customers through the [Crunchy Customer Portal](#).

If you install the below available packages with RPM, you can continue reading at the [Setup](#) section.

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

Available Packages

Without Crunchy Data Packages For installations without using packages provided by Crunchy Data, we recommend using the repository maintained at <https://github.com/lest/prometheus-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 assumes to be using Prometheus 2.x. We recommend to always use the latest minor version of Prometheus.

User and Configuration Directory Installation You will need to create a user named `ccp_monitoring` which you can do with the following command:

```
sudo useradd ccp_monitoring
```

Create a folder in `/var/lib/` and set its permissions as such:

```

sudo mkdir /var/lib/ccp_monitoring
sudo chmod 0700 /var/lib/ccp_monitoring
sudo chown ccp_monitoring /var/lib/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 `/var/lib/ccp_monitoring/prometheus` and owned by the `ccp_monitoring` user. You can set it up with:

```

sudo mkdir /var/lib/ccp_monitoring/prometheus
sudo chmod 0700 /var/lib/ccp_monitoring/prometheus
sudo chown 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
crunchy-prometheus-service-el7.conf	<code>/etc/systemd/system/prometheus.service.d/crunchy-prometh</code>
sysconfig.prometheus	<code>/etc/sysconfig/prometheus</code>
crunchy-prometheus.yml	<code>/etc/prometheus/crunchy-prometheus.yml</code>
auto.d/ProductionDB.yml.example	<code>/etc/prometheus/auto.d/ProductionDB.yml.example</code>
crunchy-alertmanager.yml	<code>/etc/prometheus/crunchy-alertmanager.yml</code>
crunchy-alert-rules.yml	<code>/etc/prometheus/crunchy-alert-rules.yml</code>

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 mkdir /var/lib/ccp_monitoring/alertmanager
sudo chmod 0700 /var/lib/ccp_monitoring/alertmanager
sudo chown 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
crunchy-alertmanager-service-el7.conf	<code>/etc/systemd/system/alertmanager.service.d/crunchy-alertmanager</code>
sysconfig.alertmanager	<code>/etc/sysconfig/alertmanager</code>

Setup

Setup on RHEL/CentOS 7

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:

- `/etc/systemd/system/prometheus.service.d/crunchy-prometheus-service-el7.conf`
- `/etc/systemd/system/alertmanager.service.d/crunchy-alertmanager-service-el7.conf`

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

- `/etc/sysconfig/prometheus`
- `/etc/sysconfig/alertmanager`

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
<code>/etc/prometheus/crunchy-prometheus.yml</code>	Modify to set scrape interval if different from the default
<code>/etc/prometheus/crunchy-alertmanager.yml</code>	Setup alert target (e.g., SMTP, SMS, etc.), receiver and route
<code>/etc/prometheus/crunchy-alert-rules.yml</code>	Update rules as needed. Default Prometheus config expects rules to be in this file
<code>/etc/prometheus/auto.d/*.yml</code>	You will need at least one file with a final <code>.yml</code> extension

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
```

Note for packaging (RHEL/CentOS 7)

The service override files must be placed in the relevant drop-in folder to override the default service files.

```
/etc/systemd/system/prometheus.service.d/crunchy-prometheus-service.conf
/etc/systemd/system/alertmanager.service.d/crunchy-alertmanager-service.conf
```

After a daemon-reload, systemd should automatically find these files and the crunchy services should work as intended.

Setup on RHEL/CentOS 6

Detailed instructions coming soon.