Postgres Workload Report

Version 1

Built at 2024-05-03T15:20:46

1 Postgres Workload Report

Postgres Workload Report (PWR) is a Python-based tool used for building PostgreSQL workload reports in HTML, Markdown, DOCX, and PDF mode. These reports mimic the reports provided by the Automatic Workload Repository (AWR) reporting tool from Oracle.

Using a Postgres connection, you can execute Postgres Workload Report from any machine with access to the server for which you want a report. Postgres Workload Report uses Lasso for collecting data, so Lasso is a prerequisite. The `edb-pwr` package has an explicit dependency on `edb-lasso` being installed.

On the Postgres server, you must have `edb_wait_states` version 1.2 or later installed and loaded. You must also create the extension, preferably on the main database, so the `edb_wait_states` functions are available for Postgres Workload Report to collect saved snapshot data.

2 Release notes

The Postgres Workload Report (PWR) documentation describes the latest version of PWR 1, including minor releases and patches. The release notes provide information on what was new in each release. For new functionality introduced in a minor or patch release, indicators in the content provide information about the release that introduced the feature.

Version	Release date
1.0.0	15 Feb 2024

2.1 Postgres Workload Report 1.0.0 release notes

Released: 15 Feb 2024

New features, enhancements, bug fixes, and other changes in Postgres Workload Report 1.0.0 include:

Type	Description
Feature	Added the `pwr execute` CLI command. The command is used to generate workload reports. While `pwr report` command requires that you run Lasso and provide its data as input, `pwr execute` takes care of an end-to-end execution, that is, it runs Lasso and `pwr report` implicitly. The `pwr execute` command requires edb-lasso (>=4.13.0) and edb_wait_states (>=1.2.0).
Feature	Added the `pwr report` CLI command. The `pwr report` command is used to generate workload reports. PWR workload reports can be written in one or more of the following formats: HTML, Markdown, PDF, and DOCX.

3 Installing Postgres Workload Report

Postgres Workload Report is provided as a Python source distributed package for all supported operating systems. You can find it in the `enterprise` and `standard` repositories.

The following command installs Postgres Workload Report and, if necessary, also pulls the `edb-lasso` package from the repository and installs it:

```
sudo <package-manager> -y install edb-pwr
```

Where `<package-manager>` is the package manager used with your operating system.

Package manager	Operating system
apt	Debian / Ubuntu
dnf	RHEL 8/9 and derivatives
yum	RHEL 7 and derivatives, CentOS 7
zypper	SLES

4 Configuring Postgres Workload Report

To reduce the number of command-line arguments needed when executing `pwr`, you can use a configuration file to specify options that always have the same value and whose values differ from the default.

Configuration file locations

Postgres Workload Report looks for a configuration file in the following places and uses the first one found:

1. The file named in the `--config` command-line option, if given.
2. The file named in the `PWR_CONFIG_FILE` environment variable, if set.
3. `~/.pwr.conf`.
4. `/etc/pwr.conf`.

The installation package creates a template for the configuration file in `/etc/pwr.conf.templ`. We recommend copying this file to one of the two places where Postgres Workload Report looks for a configuration file by default (`~/.pwr.conf` and `/etc/pwr.conf`), and editing the options in the template as necessary.

> **Note**
>
> If no configuration file is found, Postgres Workload Report assumes the default value for all options, which you can still override using the corresponding command-line options. See Using Postgres Workload Report for more on using command-line options.

> **Note**
>
> Although the configuration file extension is `.conf`, its content must be in **valid** YAML format.

Configuration file options

`input_dir`

An existing directory where the `edb_wait_states` portion of a Lasso report is located. This option is used mainly for `pwr report` execution (see Using Postgres Workload Report).

`output_dir`

Location of the directory where Postgres Workload Report writes report files. Executing `pwr` creates this directory if it doesn't exist.

`report_name`

The name of the report files generated. Usually, you specify this option on the command line because different reports typically have different names.

Don't include a file extension. An appropriate extension is added based on the output formats specified on the command line (that is, `--pdf` adds `.pdf`, `--html` adds `.html`, and so on).

`log_file`

The full path to the file where Postgres Workload Report writes the `stdout` and `stderr` logs.

`log_level`

The logging level to use when running Postgres Workload Report. The following are valid values, listed from more verbose to less verbose:

- `DEBUG`
- `INFO` (default if not specified)
- `WARNING`
- `ERROR`
- `CRITICAL`

See the Python logging documentation for more information about log levels.

`log_format`

The format of the log messages that are written to the log file. See the Python logging documentation for more information on log formatting.

`assets_dir`

The directory containing the Jinja templates used to format the HTML output and the CSS used for PDF output. The default value is `/usr/share/pwr/assets`, which contains the assets provided with the `edb-pwr` package.

5 Using Postgres Workload Report

Prerequisites

Postgres Workload Report can provide reports for Postgres servers only where the `edb_wait_states` extension, version 1.2 or later, is loaded. Furthermore, PWR can provide query wait reports only for intervals of time when the `edb_wait_states` extension was loaded on the server.

For more information, see `edb_wait_states`.

Source information for reports

After the `edb-pwr` and `edb-lasso` packages are installed on the machine, and the server has been running with the `edb_wait_states` extension loaded for a long enough period of time, you can use Postgres Workload Report to extract reports of wait states for the queries that were running during the interval of time specified.

Alternatively, Postgres Workload Report can generate reports from an existing Lasso report, assuming the report was executed on a server with `edb_wait_states` loaded. For this reason, Postgres Workload Report has a mandatory first argument, which can be either of the following:

- `execute` — Performs end-to-end execution. It calls `lasso` with appropriate options and uses the generated tarball report as the input to generate an HTML, Markdown, DOCX, or PDF report.
- `report` — Uses the directory for the stored `lasso` report contents as input for processing and generating the wait states report. Before you can use the `report` argument, a decompressed and untarred `lasso` report must already exist.

Example for the `execute` option

This example generates a report on query waits for the Postgres server `myserver` for the interval between January 10th at 9:00 and January 10th at 13:00. An incident happened around that time that must be investigated to provide a root cause. The main database on the server where `edb_wait_states` is installed is `my-oltp`.

To get the report in HTML format, use the following command:

```
pwr execute --host-name myserver --sampling-start '2024-01-10 09:00:00+00:00' \
    --sampling-end '2024-01-10 13:00:00+00:00' --html \
    --report-name 'Jan10_incident' my-oltp postgres
```

> **Note**
>
> `--sampling-start` and `--sampling-end` accept both timestamps with or without time zone. If no time zone is explicitly set in the timestamp(s), PWR uses the system time zone.

Run `pwr execute -h` to get the full list of options available.

Example for the `report` option

In some cases, you already have a Lasso report and want PWR to use the Lasso report as the source and build a report based on it. For these cases, the `pwr report` option is useful.

This example uses the same scenario already described but uses a Lasso report that was executed using the time boundaries in the previous example. Suppose that the Lasso report's name is `edb-lasso-Jan10-incident.tar.bz2` and is located in the home directory of the machine where `pwr report` will be executed.

The following commands generate an HTML report saved in `~/pwr_output/Jan10_incident.html`:

```
 cd ~/
mkdir -p pwr_tmp/
tar jxf ../edb-lasso-Jan10-incident.tar.bz2 -C ~/pwr_tmp/ --strip-components=1
pwr report \
    --input-dir ~/pwr_tmp/postgresql/dbs/my-oltp/edb_wait_states/ \
```

```
    --html --output-dir ~/pwr_output/ --report-name 'Jan10_incident'
```

Run `pwr report -h` to get the full list of options available.