



Postgres Workload Report

Release 1.0.0

EnterpriseDB Corporation.

Feb 20, 2024

Contents

1	Release notes	3
1.1	Postgres Workload Report 1.0.0 release notes	3
2	Installing Postgres Workload Report	5
3	Configuring Postgres Workload Report	7
3.1	pwr configuration file	7
3.1.1	input_dir	7
3.1.2	output_dir	7
3.1.3	report_name	8
3.1.4	log_file	8
3.1.5	log_level	8
3.1.6	log_format	8
3.1.7	assets_dir	8
4	Using Postgres Workload Report	9
4.1	Prerequisites	9
4.2	Source information for reports	9
4.3	Example for the <code>execute</code> option	10
4.4	Example for the <code>report</code> option	10

Postgres Workload Report (PWR) is a Python-based tool used for building PostgreSQL workload reports in HTML, Markdown, DOCX, and PDF mode, mimicking 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 should have `edb_wait_states` version 1.2 or later installed and loaded. The extension also must be created, preferably on the main database, so the `edb_wait_states` functions are available for Postgres Workload Report to collect saved snapshot data.

For more information about PWR, see:

- *[Installing Postgres Workload Report](#)*
- *[Configuring Postgres Workload Report](#)*
- *[Using Postgres Workload Report](#)*

CHAPTER 1

Release notes

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

Version	Release Date
<i>Postgres Workload Report 1.0.0 release notes</i>	15 Feb 2024

1.1 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	Add <i>pwr execute</i> CLI command. The command is used to generate workload reports. While <i>pwr report</i> command requires the user to run Lasso and provide its data as input, <i>pwr execute</i> takes care of an end-to-end execution, i.e., it runs Lasso and <i>pwr report</i> implicitly. The <i>pwr execute</i> command requires: edb-lasso ($\geq 4.13.0$), edb_wait_states ($\geq 1.2.0$). See PWR documentation for more details on the Postgres server requirements as well as for the workload report output formats.
Feature	Add <i>pwr report</i> CLI command. The <i>pwr report</i> command is used to generate workload reports. PWR workload reports can be written in one or more among the following formats: HTML, Markdown, PDF and DOCX. See PWR documentation for more details on the <i>pwr report</i> command.

CHAPTER 2

Installing Postgres Workload Report

Postgres Workload Report is provided as a Python source distributed package for all supported operating systems. It can be found 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 too.

To install Postgres Workload Report, enter the following command:

```
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

Configuring Postgres Workload Report

3.1 pwr configuration file

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 but that value differs from the default.

During execution, `pwr` looks for an existing configuration file in `~/.pwr.conf` and `/etc/pwr.conf`, in that order using the first one found. However, if the `--config` option specifies a configuration file, that overrides the default locations.

The installation package creates a template for the configuration file in `/etc/pwr.conf.template`. We recommend copying this file to one of the two places where `pwr` looks for a configuration file and editing the options in the template as necessary.

The following sections describe the options you can configure.

3.1.1 input_dir

Identifies an existing directory where the `edb_wait_states` contents of a Lasso report are located. This option is used mainly for `pwr report` execution (see *Using Postgres Workload Report*).

3.1.2 output_dir

Specifies where reports are outputted. During execution, Postgres Workload Report tries to create the directory if it doesn't exist.

3.1.3 report_name

Provides the name of the report files generated. Usually, this option is specified from the command line because different reports typically have different names.

3.1.4 log_file

Provides the full path to the file where `pwr` writes the `stdout` and `stderr` logs.

3.1.5 log_level

Specifies the logging level to use when running Postgres Workload Report. Valid values are, 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.

3.1.6 log_format

Provides the format of the log messages that are written to the log file. See [the Python logging](#) documentation for more information on log formatting.

3.1.7 assets_dir

Identifies the directory where you can find the jinja templates used to format the HTML output and the CSS used for PDF output. Typically, the directory you may want to use by default is `/usr/share/pwr/assets`, which contains the assets provided with the `edb-pwr` package.

Using Postgres Workload Report

4.1 Prerequisites

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

For more information, see [edb_wait_states](#).

4.2 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 two:

- `execute`
- `report`

The `execute` argument performs end-to-end execution, calling `lasso` with appropriate options and using the generated tarball report as the input to generate an HTML, Markdown, DOCX, or PDF report.

The `report` argument uses the directory for stored `lasso` report contents as input for processing and generating the wait states report. To use the `report` argument, a decompressed and untarred `lasso` report must already exist.

4.3 Example for the execute option

For this example we want to generate 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 which 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 such a report in HTML format we would use the following command:

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

We can run `pwr execute -h` to get the full output of options available.

4.4 Example for the report option

In some cases you may already have a Lasso report and would like 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.

For this example we can use the same scenario already described but we use a Lasso report that has already been executed using the time boundaries shared above. Let's suppose that the Lasso report's name is `edb-lasso-Jan10-incident.tar.bz2` and it is located in the home directory of the machine where `pwr report` will be executed.

The following command executions 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
```

We can run `pwr report -h` to get the full output for the options available.