Automating audits with AutoWebPerf

- What is AWP for?
- Architecture overview
- Automating audits with AWP
  - Recurring audits
- Visualising audit results in Data Studio {: #data-studio }
- What's next?

AutoWebPerf (AWP) is a modular tool that enables automatic gathering of performance data from multiple sources. Currently there are many tools available to measure website performance for different scopes (lab and field), such as Chrome UX Report, PageSpeed Insights, or WebPageTest. AWP offers integration with various audit tools with a simple setup so you can continuously monitor the site performance in one place.

The release of Web Vitals guidance means that close and active monitoring of web pages is becoming increasingly important. The engineers behind this tool have been doing performance audits for years and they created AWP to automate a manual, recurring, and time consuming part of their daily activities. Today, AWP has reached a level of maturity and it's ready to be shared broadly so anyone can benefit from the automation it brings.

The tool is accessible on the AutoWebPerf public repository on GitHub.

What is AWP for?

Although several tools and APIs are available to monitor the performance of web pages, most of them expose data measured at a specific time. To adequately monitor a website and maintain good performance of key pages, it's recommended to continuously take measurements of Core Web Vitals over time and observe trends.

AWP makes that easier by providing an engine and pre-built API integrations which can be programmatically configured to automate recurrent queries to various performance monitoring APIs.

For example, with AWP, you can set a daily test on your home page to capture the field data from CrUX API and lab data from a Lighthouse report from PageSpeed Insights. This data can be written and stored over time, for example, in Google Sheets and then visualised in the Data Studio dashboard. AWP automates the heavy-lifting part of the entire process, making it a great solution to follow lab and field trends over time. See Visualising audit results in Data Studio below for more details).

Architecture overview

AWP is a modular-based library with three different types of modules:

- the **engine**
- **connector** modules
- **gatherer** modules

The engine takes a list of tests from a connector (for example, from a local CSV file), runs performance audits through selected gatherers (such as PageSpeed Insights), and writes results to the output connector (for example, Google Sheets).

A diagram of AWP's architecture.

AWP comes with a number of pre-implemented gatherers and connectors:

- Pre-implemented gatherers:
  - CrUX API
  - CrUX BigQuery
  - PageSpeed Insights API
  - WebPageTest API
- Pre-implemented connectors:
  - Google Sheets
  - JSON
  - CSV

Automating audits with AWP

AWP automates the performance audits via your preferred audit platforms such as PageSpeed Insights, WebPageTest, or CrUX API. AWP offers the flexibility to choose where to load the list of tests, and where to write the results to.

For example, you can run audits for a list of tests stored in a Google Sheet, and write the results to a CSV file, with the command below:

```
PSI_APIKEY=<YOUR_KEY> SHEETS_APIKEY=<YOUR_KEY> ./awp run sheets:<SheetID> csv:output.csv
```

Recurring audits

You can run recurring audits in daily, weekly, or monthly frequency. For example, you can run daily audits for a list of tests defined in a local JSON like below:

```
{
  "tests": [
    {
      "label": "web.dev",
      "url": "https://web.dev",
      "gatherer": "psi"
    }
  ]
}
```

The command below reads the list of audit tests from the local JSON file, runs audits on a local machine, then outputs results to a local CSV file:

```
PSI_APIKEY=<YOUR_KEY> ./awp run json:tests.json csv:output.csv
```

To run audits every day as a background service continuously, you can use the command below instead:

```
PSI_APIKEY=<YOUR_KEY> ./awp continue json:tests.json csv:output.csv
```

Alternatively, you can set up the crontab in a Unix-like environment to run AWP as a daily cron job:

```
0 0 * * * PSI_APIKEY=<YOUR_KEY> ./awp run json:tests.json csv:output.csv
```

You can find more ways to automate daily audits and result collection in the AWP GitHub repository .

Visualising audit results in Data Studio {: #data-studio }

Along with continuously measuring Core Web Vitals, it is important to be able to evaluate the trends and discover potential regressions with real user metrics (RUM) or the Chrome UX Report (CrUX) data collected by AWP. Note that Chrome UX Report (CrUX) is a 28-day moving aggregation, hence it is recommended to also use your own RUM data along with CrUX so you can spot regressions sooner.

Data Studio is a free visualization tool that you can easily load performance metrics into and draw trends as charts. For example, the time series charts below show Core Web Vitals based on Chrome UX Report data. One of the charts shows increasing Cumulative Layout Shift in recent weeks, which means regressions in the layout stability for certain pages. In this scenario, you would want to prioritize the efforts to analyze the underlying issues of these pages.

A screenshot of Core Web Vitals results in Data Studio.

To simplify the end-to-end process from data collection to visualization, you can run AWP with a list of URLs to automatically export results to Google Sheets with the following command:

```
PSI_APIKEY=<YOUR_KEY> SHEETS_APIKEY=<YOUR_KEY> ./awp run sheets:<SheetID> csv:output.csv
```

After collecting daily metrics in a spreadsheet, you can create a Data Studio dashboard that loads the data directly from the spreadsheet, and plots the trends into a time series chart. Check out Google Spreadsheets API Connector for detailed steps about how to set up AWP with spreadsheets as a data source to visualize on Data Studio.

What's next?

AWP provides a simple and integrated way to minimize the efforts to set up a continuous monitoring pipeline to measure Core Web Vitals and other performance metrics. As for now, AWP covers the most common use cases and will continue to provide more features to address other use cases in the future.

Learn more in the AutoWebPerf repository.

□□□ #3
□ □□□ □□□ 2026-10-31 21:39:00 UTC
□ cocobook-sync □□□ 2026-11-04 17:36:34 UTC