

Google Lighthouse Batch Pull Instruction Guide

Written and Developed by: Connor Shocrylas, SEO Strategist

Version 1.0

# Batch Pulling Lighthouse Reports

Google Lighthouse is a fantastic resource that enables users to pull detailed speed reports of their sites. Reports are broken into 5 different categories each assigned a score 0 out of 100:

* Performance
  + Provides reporting/scoring on which areas of your individual page could be optimized for speed.
* Progressive Webb App (PWA)
  + Provides a list of validation checks that determine whether your site is following [PWA baselines](https://developers.google.com/web/progressive-web-apps/checklist).
* Accessibility
  + Highlights opportunities to [improve the accessibility](https://developers.google.com/web/fundamentals/accessibility/) of your websites pages.
* Best Practices
  + Documents simple web development best practices.
* SEO
  + Checks for basic SEO best practices, but is still unable to check for most items.

Lighthouse can be run via your web browser by simply navigating to Inspector 🡪 Audit and run on a page-by-page basis. As it only runs on one URL at a time, it can be a time-consuming process to be able to pull multiple Audit Reports. The following guide solves this issue by moving the audit process to your local machine and enabling SEOer’s to batch pull Lighthouse reports and have a more encompassing POV.

**Initial Test Results**

* **Site Tested:** <https://www.pulte.com>
* **Average Page Speed:** Slow to Medium
* **Time Taken to Manually Audit 10 Pages via Browser:** 30 Minutes
* **Time Taken via Batch Solution:** 8 Minutes
* **Total Time Savings:** 24 Minutes
* **Percent of Improvement:** 74%

Running a test on Pulte’s website proved that this solution will save you a drastic amount of time. As Pulte’s site is slow it takes a while to manually navigate to each page, enter the inspector and setup your audit. Whereas, the Lighthouse batch solution showed **time improvements of ~74%.** As this solution can also provide exports in HTML, JSON and CSV it also saves time on the latter end of data analysis/consumption.

Requirements to Run

* Windows Machine
* Command Line Interface
* NodeJS
* Lighthouse Node Module
* Text Editor

## Installing GitBash & Using Command-Line Interface (CLI)[[1]](#footnote-1)

To make running this simple script as painless as possible, it is recommended to install a CLI that takes advantage of Unix based commands vs. Windows. You can run this program using the basic Command Prompt that comes with Windows, but it is a lot easier to use a different Shell.

**Steps:**

* Visit <https://git-scm.com/download/win>
* Open GIT Executable file and start installation
* Click “Next” for all Window prompts

Once installed press the Windows key on your keyboard and type Git” then select the program from the start menu.

After clicking the program from the start menu, you should see your brand new CLI.

## Installing NodeJS

Once GitBash is installed, or you have your preferred CLI already, it’s time to install NodeJS on your local machine. NodeJS is a popular asynchronous even driven JavaScript runtime which essentially means that it executes JavaScript outside of a browser. As JavaScript is a Client-Side Scripting language, NodeJS makes it so that developers can execute it at the server-side – thus turning JavaScript into a Server-Side Scripting language.

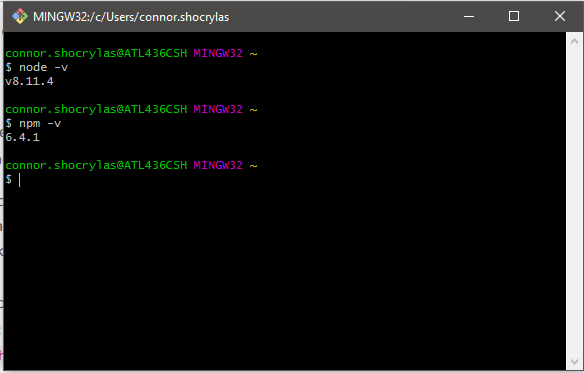
Steps to Install

* Visit <https://nodejs.org/en/download/>
* Download the Windows Installer (.msi) “LTS” version and install the 32-bit or 64-bit version depending on your OS Type
  + You can check your OS type by pressing the windows key and typing “System Information”
  + Clicking the top result will show a line that says, “System type:” followed by your OS version
* Follow all installer prompts

### Testing NodeJS Install

At this point you have installed a new CLI and should have NodeJS on your machine. Now is the time to test it all.

Open your new CLI by pressing the Windows key and typing in “Git”. Once opened, type the following commands node -v. If you have followed all the step in this document the CLI should return the version number of Node on your machine. Next type npm -v which should return a similar result.



## Installing Lighthouse

Now it’s time to install the [Google Lighthouse Node Module](https://www.npmjs.com/package/lighthouse).

Within your CLI type the following command npm install -g lighthouse and let the window run. Once the CLI is commands have completed type lighthouse -v. To verify the install, type lighthouse -–help which should print out a list of Lighthouse commands you can use.

### Testing Lighthouse Install

With lighthouse installed you can now run page audits right within your CLI instead of having to navigate to each individual page. Let’s try running a simple lighthouse audit.

In your CLI type the following commands:

ls

cd Desktop

mkdir Test

cd Test

lighthouse <https://360i.com/>

After pressing enter for the last line, a Google Chrome window should open on your computer and the CLI will be having different commands run on it.

What did we just do?

* We navigated to our Desktop using the “cd” command
* We created a new folder on our Desktop called “Test”
* We navigated to our new folder
* We ran a lighthouse audit for 360i’s website

Accessing the folder from your desktop will reveal an HTML file that when opened will show you the audit for 360i’s homepage.

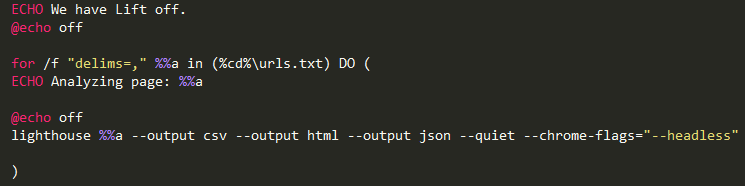
Congrats! You have successfully run your first CLI Lighthouse audit.

Now it’s time to run multiple audits at once.

## Preparing for Batch Reports

To pull multiple reports at once we need to write a bit of code. Out of the box, the Lighthouse node module doesn’t expand our ability to run multiple audits – albeit it does provide us with more customization. To get Lighthouse to run multiple audits we need to write a recursive script that runs our commands multiple times.

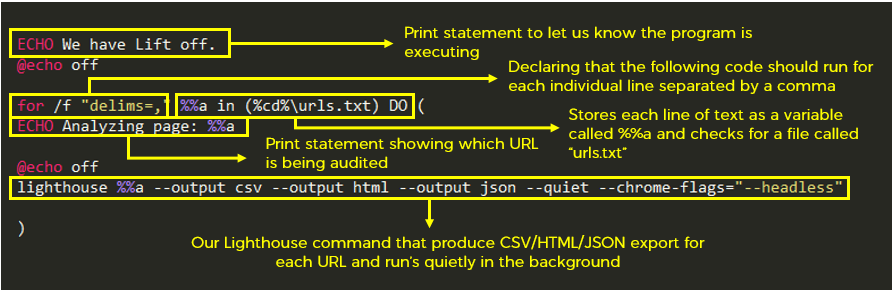
For the first prototype, I have written a simple Batch script that does exactly this:



The program above is a simple DO Loop that runs a lighthouse Node command for every URL that is found in the file “urls.txt”. To setup the batch pull report, you need three files which can be downloaded from the [360i Agency Email Google Drive](https://drive.google.com/open?id=1xJgeusRzEqf8A05TzCIFZkghgXxrfsJn).

Download those files and place them in your already created “Test” folder or simply create a new one.

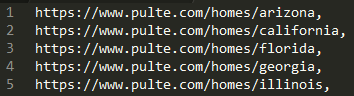
### Dissecting the Code



### Preparing Your URLs

Once those files are placed in your desired folder you can begin modifying the “urls.txt” file which simply contains all the URLs you wish to batch pull for. I recommend **auditing no more than 50 URLs at a time** as that has proved to be the cap for optimal speed.

Make sure that each URL is on a separate line and has a comma at the end for this program to run correctly and that you have the full protocol at the front (i.e. https:// or http://).



### Running the Script

Open your CLI and navigate to the folder that contains the files. In your command prompt type, the following:

cmd

test.bat

At this point you should see We have Lift off. printed on your screen and the program is running its course.

After it is done running, you should have a HTML/CSV/JSON file for each individual audit. You can run these through a combiner tool or a JSON reader to compile and read your results.

### Combining CSVs

As an added bonus to the batch pull report, I have written a simple script that combines all the CSV files in a folder into one document. Within your CLI, navigate to your project folder where your Lighthouse CSV files live. In there type the following commands:

cmd

combine.bat

Depending on how many CSV files you have, this can take either a couple seconds or a couple minutes. The final product should be a new file in your project folder called “combine.csv”.

## Future Additions[[2]](#footnote-2)

* Migrating to Node App or Python Script
* CSV Combining
* Custom File Naming
* Web Speed Dashboard
* Client Customized Speed Reports

1. This is an extra step and not required. Move to Installing NodeJS if you do not wish to install GitBash. [↑](#footnote-ref-1)
2. Just a couple ideas I have currently. [↑](#footnote-ref-2)