

# How Fast Does Your Website Need to Be?

Data, rules, and methodologies to help benchmark and improve web and mobile performance

Customers are always interested in improving the speed and performance of their web and mobile apps. In many cases, they've already spent time, effort, and money to move to the cloud. They've also fundamentally changed the way their teams deliver software, and might even use Kubernetes to scale faster. They face tons of complexity already, which is why they'd prefer simple answers to the inevitable question about web performance: "How fast do we need to be?"

Our answer is usually the same: "It depends."

Admittedly, that's not a response anyone wants to hear. Unfortunately, the truth is that—while uptime is easy to quantify—there is no single metric to clearly define page performance.

Studies point to the fact that "faster is better." Gomez, the web performance division of Compuware, found that after three seconds of waiting, up to [40% of users](#) abandon a site. Additionally, Walmart found that improving page speed by even 100ms saw [revenue increase 1%](#). However, while speed is a critical component, it's only one part of a dynamic page experience.

Customers navigate your site on a dizzying combination of web browsers (and their different versions), devices (both new and old),

## General Rule #1: Use more than one metric to measure page performance

To really understand how customers experience your pages, you need to measure a few important themes of user experience. Choose a few categories from different components of your page experience. Listed below are some of those categories, with New Relic's metrics and their definitions:

### Page Load Experience

- **First paint (FP):** How long until the first pixel of a background renders
- **First Contentful Paint (FCP):** How long until content begins to render
- **Largest Contentful Paint (LCP):** How long until the largest image or video begins to render

### User Interactivity

- **First Interaction:** Measures when a user first engages with your page
- **First Input Delay (FID):** The time from when the user engages until the web browser processes the interaction

### Visual Stability

- **Cumulative Layout Shift (CLS):** Measures how end users experience sudden unexpected shifts from your content dynamically resizing
- **Long Tasks:** Measures JavaScript code that blocks the main thread

and networks (from high-speed to sluggish hotel internet). Each person tolerates speed and performance in their own way. Luckily for us, we have data and some general rules to inform the optimization process.

While “it depends” can be frustrating to hear, it’s a great opportunity to explore web page fundamentals, customer happiness, and creating a performance-focused culture throughout a digital business. Here’s New Relic’s approach to page speed and performance using [Full-Stack Observability](#), some general rules to follow, as well as a step-by-step guide to creating your own dashboards to keep tabs on your digital properties.

## The right metrics to monitor

There’s no single metric to define page performance. For years, there’s been a prolonged reliance on page load time (window.onload) as a catchall for measuring and reporting end-user performance. While that metric was useful for Web 1.0, it’s often misleading when trying to understand modern pages.

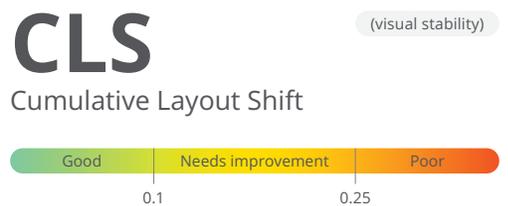
To more precisely measure performance, it’s helpful to capture metrics that reflect how users perceive the experience your pages deliver. User-centric perceived performance metrics help site owners understand key components of their page experience, such as how fast they deliver visual content, how fast their pages respond to user interaction, and how their page creates a smooth experience when loading.

### Google’s Core Web Vitals

We recommend standardizing on [Google’s Core Web Vitals](#) for selecting and benchmarking performance. Google recommends three measurements to define page health: Largest Contentful Paint (LCP), First Input Delay (FID), and Cumulative Layout Shift (CLS). The corresponding rankings of “Good,” “Needs Improvement,” and “Poor,” are shown alongside each measurement. For each measurement, Google recommends the following thresholds for “good,” “needs improvement,” and “poor” performance across the 75th percentile of your users. New Relic offers full support for Google’s Core Web Vitals. Here’s how to use our [PageViewTiming event](#) to create dashboards for them.

## How to visualize your page performance with dashboards

New Relic is [all about your data](#). Our Real User Monitoring agent offers [eight total event types](#), each with dozens of attributes to help measure, benchmark, and alert on JavaScript errors, single-page appli-



cation (SPA) pages' details, AJAX requests, spans in distributed tracing, geographical performance, and more. For load timing data specifically, we have two major event types: [PageViewTiming event](#) and [Browser-Interaction](#). Both have the same set of default attributes, but Browser-Interaction has extra features that include AJAX response times and route changes.

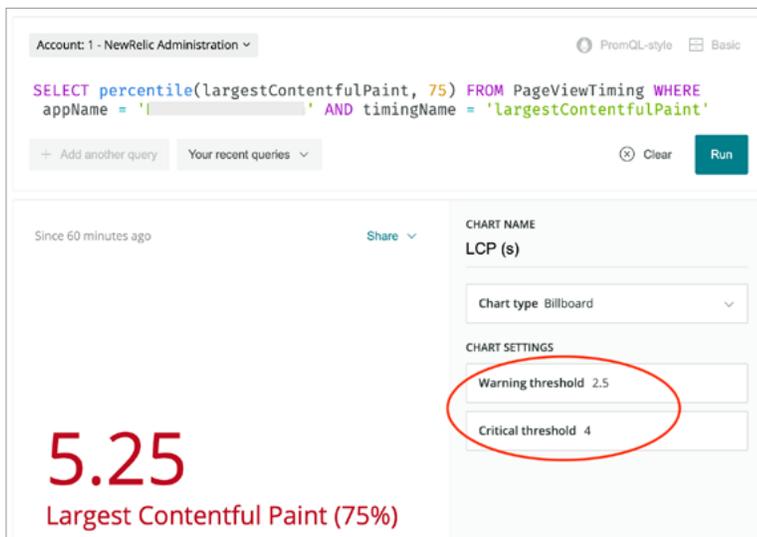
Let's use our PageViewTiming event to dashboard all three Core Web Vitals and visualize page performance.

### Largest Contentful Paint

1. Use this NRQL query:

```
SELECT percentile(largestContentfulPaint, 75) FROM PageViewTiming WHERE appName = 'insert your app name' AND timingName = 'largestContentfulPaint'
```

2. Add thresholds recommended by Google's Core Web Vitals:



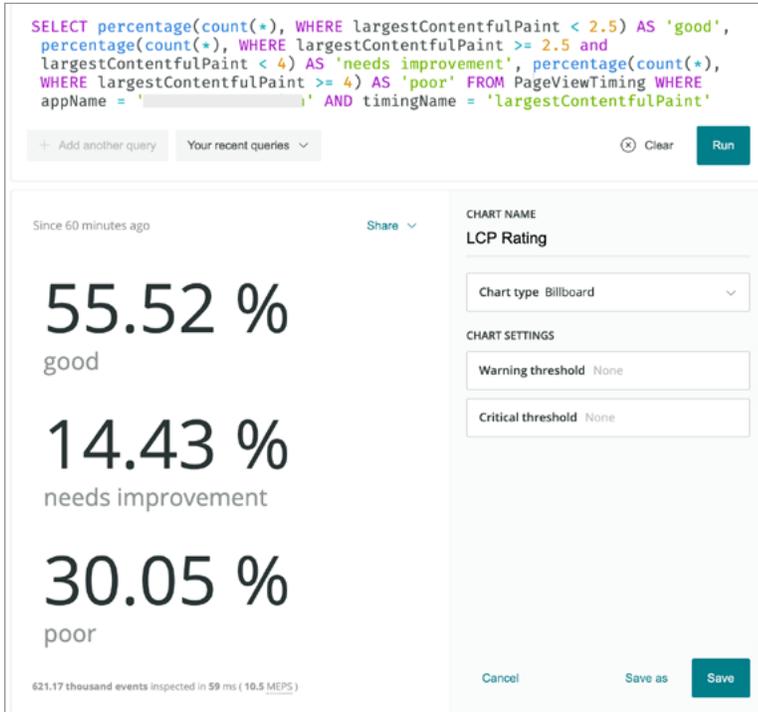
(Note: We need to load large images much faster, because we've passed the "poor" threshold.)

3. Quantify good, tolerable, or poor experiences relative to loading visual content:

```
SELECT percentage(count(*), WHERE largestContentfulPaint < 2.5) AS 'good', percentage(count(*), WHERE largestContentfulPaint >= 2.5 and largestContentfulPaint < 4) AS 'needs improvement', percentage(count(*), WHERE largestContentfulPaint >= 4) AS 'poor' FROM PageViewTiming WHERE appName = 'insert your app name' AND timingName = 'largestContentfulPaint'
```

### General Rule #2: Measure backend vs. frontend

For troubleshooting, it's helpful to quickly compare server-side and client-side performance. Because New Relic connects your [full stack](#), we recommend adding TimeToFirstByte (TTFB) (length of time from user's navigation to first byte of the response) to measure web server performance. "Good" TTFB performance is generally .2 seconds (200ms) or faster, and "poor" is generally slower than 1 second. Standardizing on TTFB + LCP + FID + CLS helps align with Google's Core Web Vitals, while also measuring backend performance.

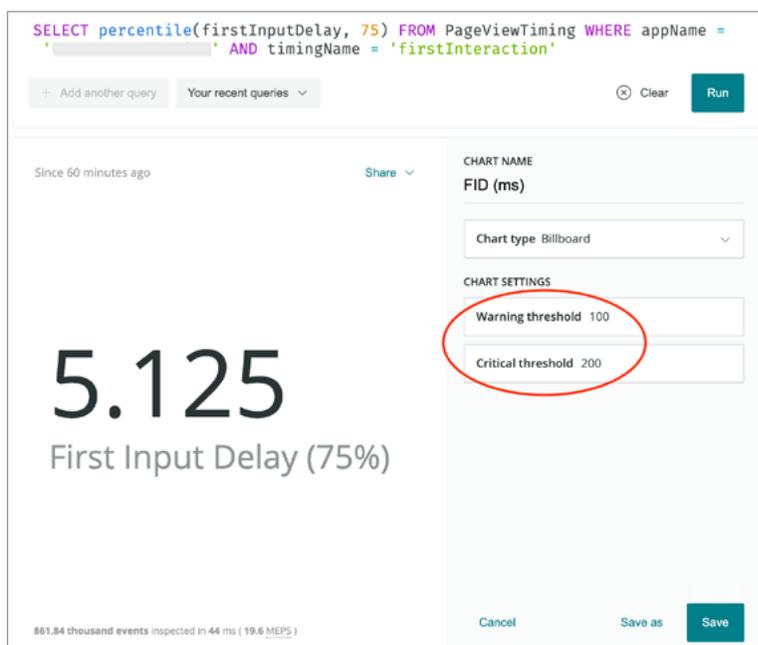


### First Input Delay

1. Use this NRQL query:

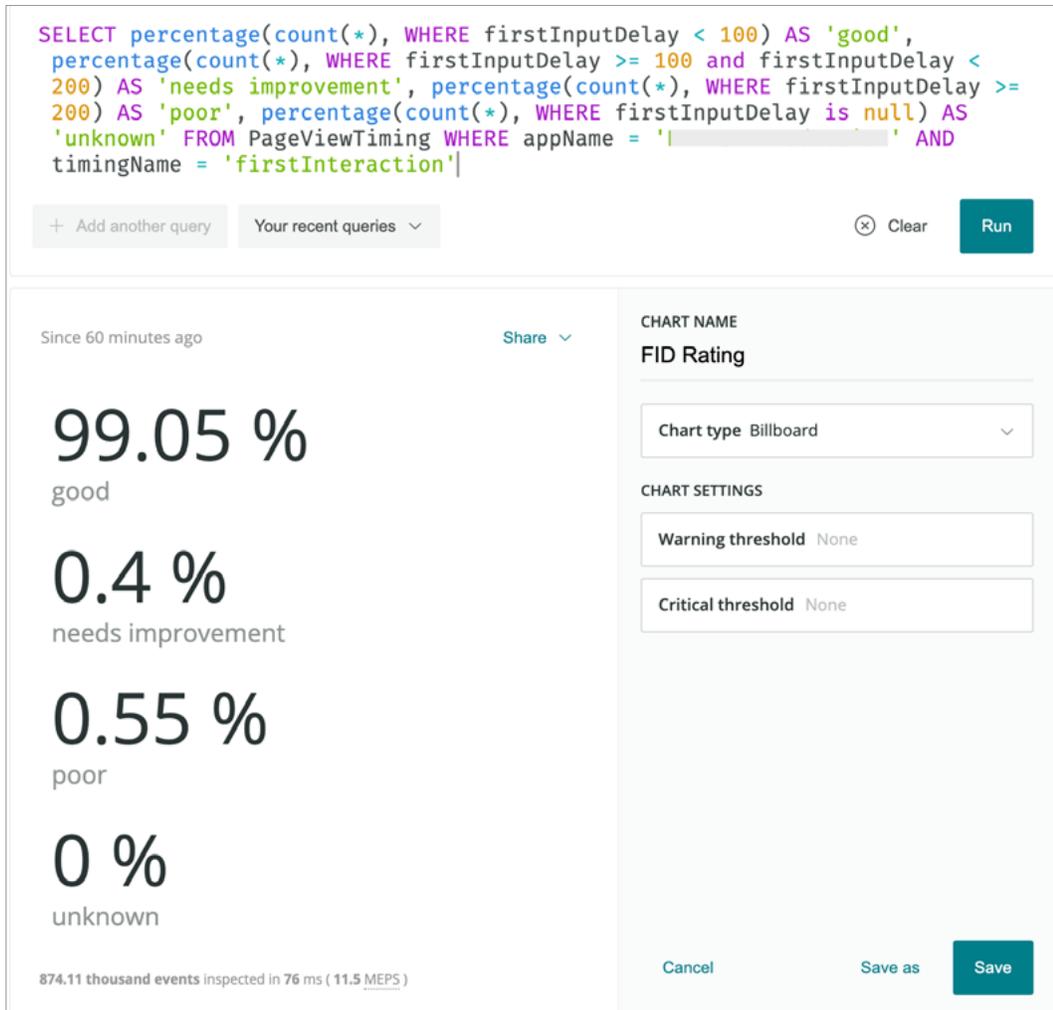
```
SELECT percentile(firstInputDelay, 75) FROM PageViewTiming WHERE appName = 'insert your app name' AND timingName = 'firstInteraction'
```

2. Add thresholds recommended by Google's Core Web Vitals:



- Quantify good, tolerable, or poor experiences relative to page interactivity:

```
SELECT percentage(count(*), WHERE firstInputDelay < 100) AS 'good',
percentage(count(*), WHERE firstInputDelay >= 100 and firstInputDelay <
200) AS 'needs improvement', percentage(count(*), WHERE firstInputDelay
>= 200) AS 'poor', percentage(count(*), WHERE firstInputDelay is null) AS
'unknown' FROM PageViewTiming WHERE appName = 'insert your app name' AND
timingName = 'firstInteraction'
```



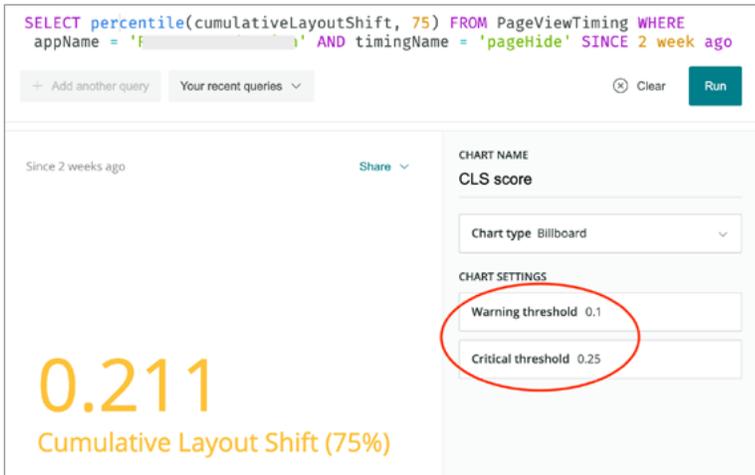
(Note: Our page is extremely fast responding to user engagement.)

## Cumulative Layout Shift

- Use this NRQL query:

```
SELECT percentile(cumulativeLayoutShift, 75) FROM PageViewTiming WHERE
appName = 'insert your app name' AND timingName = 'pageHide' SINCE
2 week ago
```

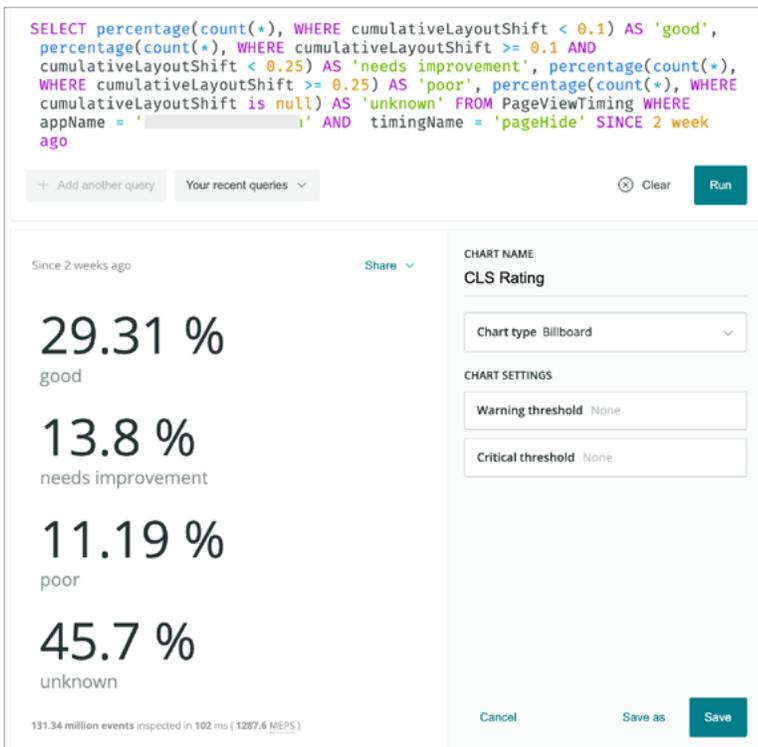
2. Add thresholds recommended by Google’s Core Web Vitals:



(Note: We need improvement; the metric is yellow because we’ve passed the warning threshold.)

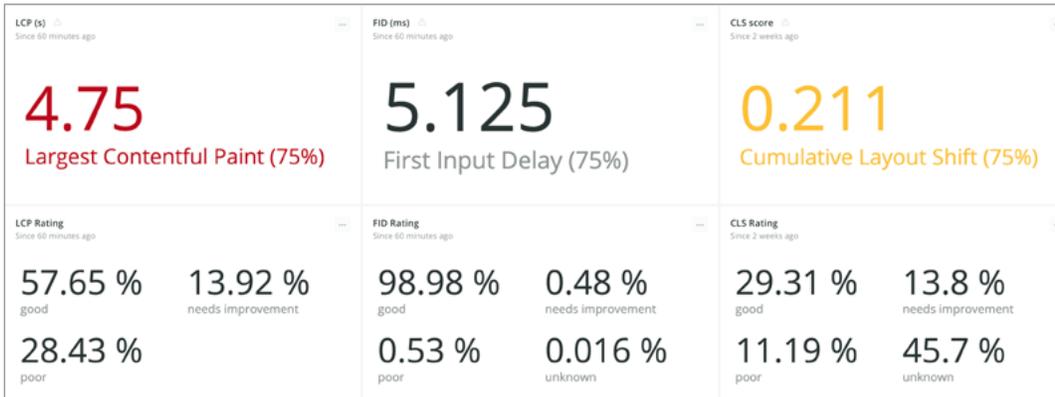
3. Find how many users are having good, tolerable, or poor experiences:

```
SELECT percentage(count(*), WHERE cumulativeLayoutShift < 0.1) AS
'good', percentage(count(*), WHERE cumulativeLayoutShift >= 0.1 AND
cumulativeLayoutShift < 0.25) AS 'needs improvement', percentage(count(*),
WHERE cumulativeLayoutShift >= 0.25) AS 'poor', percentage(count(*), WHERE
cumulativeLayoutShift is null) AS 'unknown' FROM PageViewTiming WHERE
appName = 'insert your app name' AND timingName = 'pageHide' SINCE
2 week ago
```



(Note: There is a segment of “unknowns” because not all web browsers provide CLS.)

View them together:



## The importance of lab and field data

So far we've talked about data from real users, otherwise known as field data. Lab data is useful for understanding optimal performance and to help find and eliminate poor performance before you deploy.

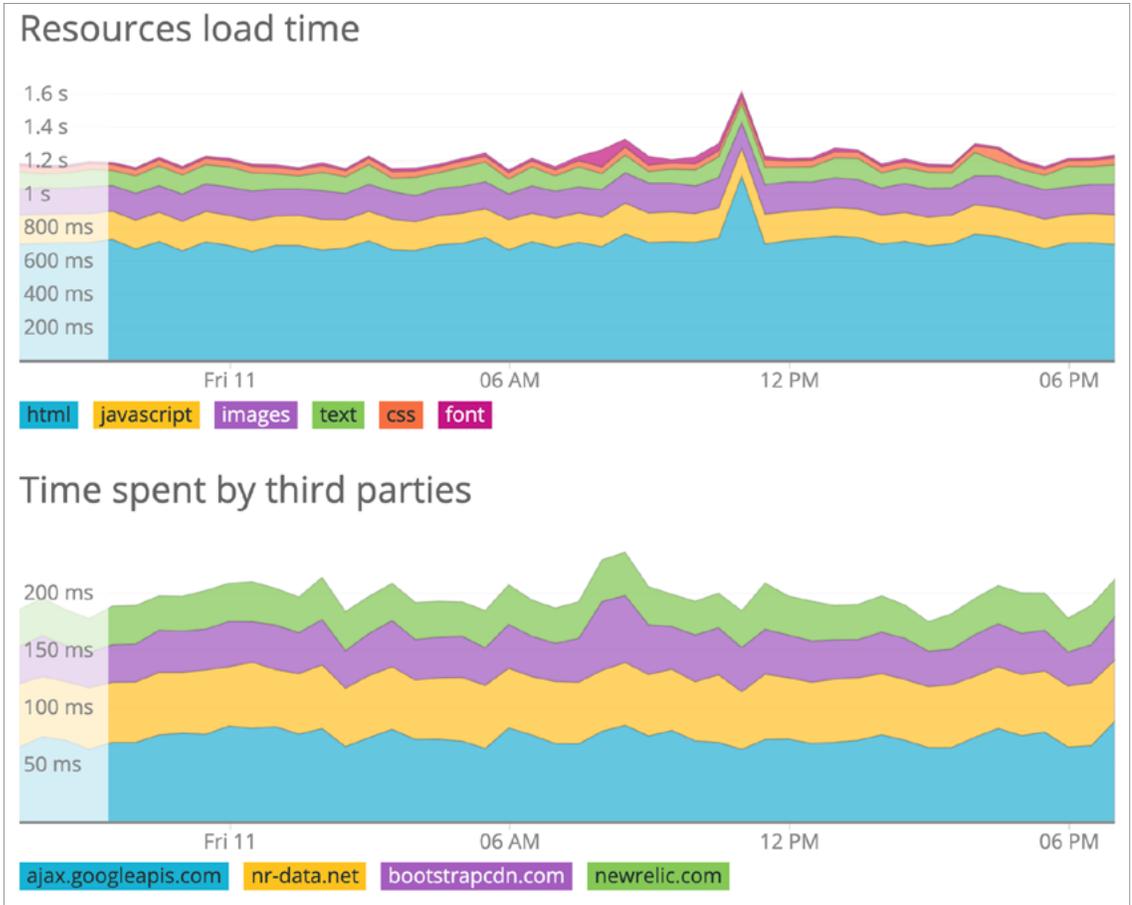
Lab data collected from synthetic monitors is useful for optimizing page performance throughout the DevOps process. It's called lab data because you're using a predefined set of conditions (typically across fast networks, with specific browsers or devices) to simulate user engagement and measure feedback. Synthetic monitors help tune performance because you can find latency and errors before releasing to production. Also, developer-focused synthetic solutions provide the ability to use JavaScript to fully recreate all page components and dependencies, therefore site operators can understand the health of their functionality and performance beyond uptime.

Modern web pages rely heavily on APIs and third parties to deliver information. Synthetics' [scripted API monitor](#) goes beyond uptime to show request/response times, connection times, and information on payload size. This helps you determine if your APIs are up, fast, and returning valid results.

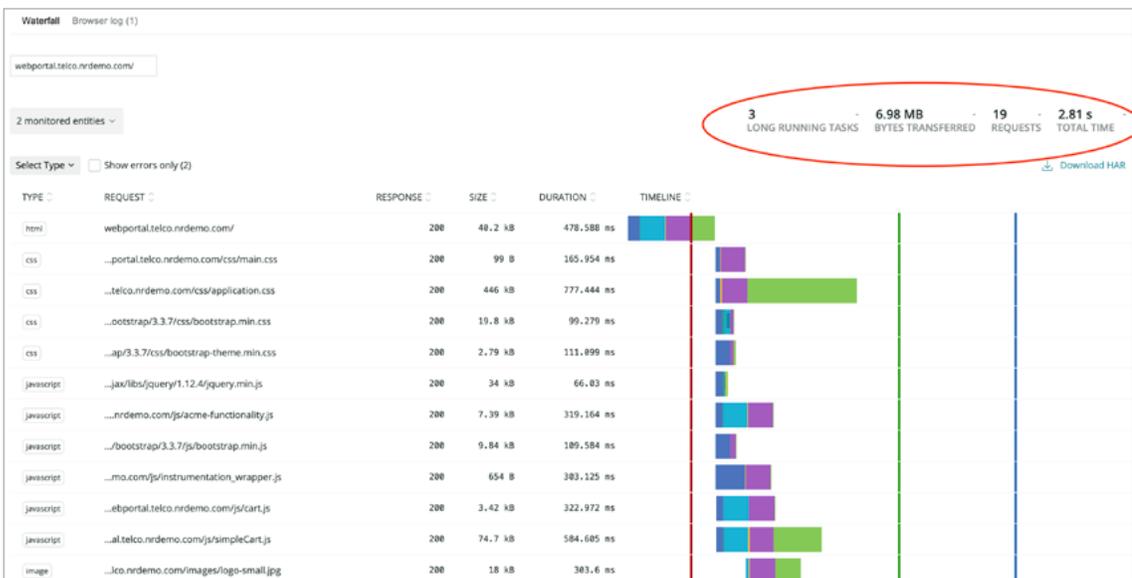
New Relic One automatically displays third-party performance in the "resources" section of each monitor, and users can leverage NRQL to further isolate performance outliers.

**General Rule #3: Be as fast, if not 20% faster, than your competition**

Given a limited amount of developers, time constraints, and the resources deemed necessary to track and measure your online business, it might not be possible to achieve a "good" score in all of Google's Core Web Vitals. The [20% rule](#) says that in order for users to barely notice a difference in time duration, it has to be changed by a minimum of 20%. Therefore, we can create synthetic monitors and leverage paint timing metrics (FP, FCP) to compare, and set objective targets for how quickly we deliver visual content vs. our competition.



To further tune performance, web developers access [Long Tasks](#), bytes transferred (aka page weight), page resource performance, and resource waterfall charts for each monitor result.



(Detailed waterfall chart of long tasks, bytes transferred, and requests made)

## Take action

**If you want to create a culture of performance, you must first start with** measuring a few core components of your page experience.

While “it depends” can be frustrating to hear—especially for managers who have enough complexity across backend services and infrastructure—it’s also an opportunity to understand how sites impact end users, and what drives their online satisfaction. “How fast do we need to be?” can also result in clearer goals and increased understanding between engineering and marketing teams, and better business results.

We encourage you to set up dashboards in alignment with Google’s Core Web Vitals and rely on their guidance for thresholds. Create a few monitors to understand how quickly you display visual content (FP, FCP) vs. your competition. Aim to be at least 20% faster. Understand the impact that APIs and third parties have on your page performance. Use New Relic and other great tools such as [Google Lighthouse](#) to find performance bottlenecks and ensure successful deployments.

Wherever you are in your page performance journey, remember that web performance is both a science and an art. It’s an examination of slivers of time in your web browser, and an exploration of human behavior. However, the most important thing to remember is that you can’t improve what you don’t measure. So jump in! Your customers are waiting (hopefully not too long).

### Experience Full-Stack Observability

See how New Relic can give your end users a better experience with your web and application properties with our Full-Stack Observability offering, which also includes APM, Infrastructure Monitoring, Logs in Context, and more.

Sign up for free at [newrelic.com/platform/full-stack-observability](https://newrelic.com/platform/full-stack-observability)

### More helpful resources

There are also lots of great resources to help inform page speed and performance. For general best practices, the folks at [web.dev](#) are your best bet. [Rick Viscom’s chapter on performance](#) from [2019’s Web Almanac](#) is a super-detailed guide to understanding “where to start.” To help embed performance into your builds and development culture, consider establishing a [performance budget](#). Make sure you follow JavaScript best practices; [The Cost of JavaScript](#) by [Addy Osmani](#) is a tremendously helpful resource. [Lara Hogan’s Designing for Performance](#) is a helpful companion for learning the basics. [Tammy Everts’](#) book, [Time Is Money](#), and her web page [WPO stats](#), help quantify the impact of web performance on business outcome.