

THE DEFINITIVE GUIDE TO

Optimizing Your Magento Site

FOR BETTER PERFORMANCE + SCALABILITY

By section.io, Magento Select Technology Partner

section.io 



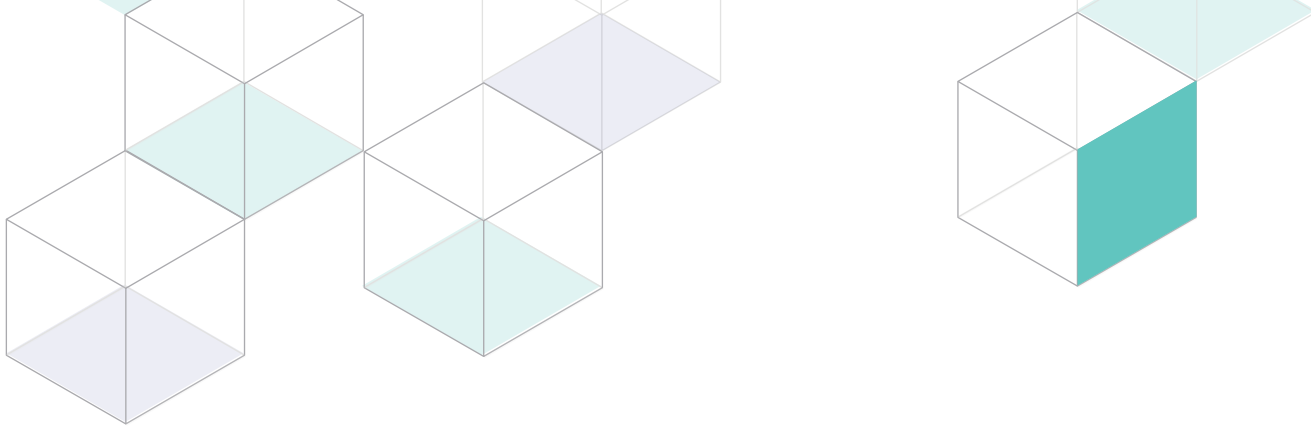


Table of Contents

- 1 | Show Me The Money**
Why Performance And Scalability Are Important For Your Magento Website 03
- 2 | Data Driven**
How To Measure Your Web Performance 07
- 3 | The Host With The Most**
Choosing The Right Hosting For You 11
- 4 | Getting To Know You**
An Introduction To Caching For Performance 15
- 5 | Digging Deeper**
How Varnish Cache And Magento Work Together 19
- 6 | The Value Of Your Network**
Considering The Benefits Of Content Delivery Networks 24
- 7 | The Meat Of It**
Programming For Performance On Magento 2 30
- 8 | Bonus Security Section**
Protecting Your Magento Website From Attacks 40
- 9 | Get Started**
Your Customizable Action Plan 45

1 | Show Me The Money

Why Performance And Scalability Are Important For Your Magento Website

Summary:

Website performance is the speed at which your web pages are downloaded and displayed to your website visitors.

Website scalability is the ability for a site to handle ever-increasing amounts of traffic and sudden bursts of traffic.

Studies show that improving both website performance and scalability results in more page views and increased ecommerce revenue.

Welcome to section.io's guide to optimizing your Magento website for better performance and scalability. We've also thrown in a bit of information about making sure your site is secure and protected from attacks in [Chapter 8](#).

If you already know why performance and scalability are so important for ecommerce sites, then feel free to skip to our more technical chapters on implementing Varnish Cache for Magento, determining if your Magento site could benefit from a Content Delivery Network, and programming Magento 2 for better performance and scalability. If you need a bit of a refresher on web performance and scalability and why they are crucial to both user experience and increased revenue, start here for a topic overview.

What Is Website Performance?

Website performance is defined as the speed at which your web pages are downloaded and displayed to your website visitors and potential ecommerce customers. There are [numerous studies from large websites such as Amazon, Google, and Firefox](#) that clearly indicate better website performance leads to more page views, a lower bounce rate, and an increase in revenue.



You may be wondering if website performance impacts small to medium-sized Magento ecommerce sites as well, and the answer is a resounding yes. For one thing, Google ranks sites with better performance more highly, giving you better Search Engine Optimization (SEO) and an increased chance that you will show up organically in search results. At section.io, we have done studies with a range of customers in different industries that demonstrate how much page speed impacts conversion rates and revenue:

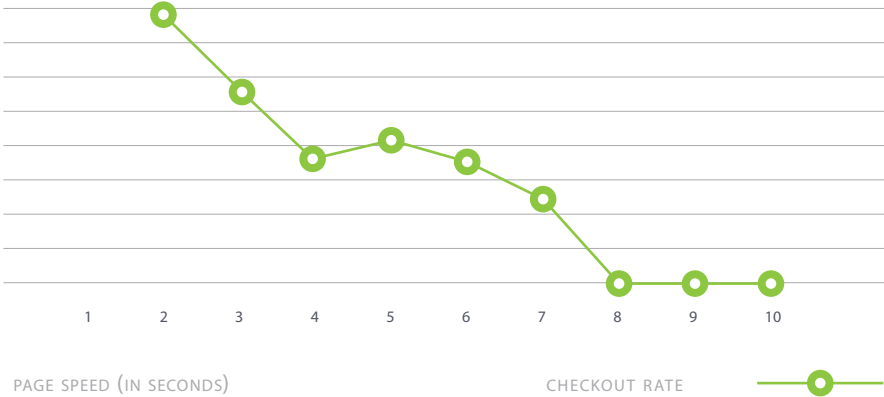
Results From section.io Studies On Ecommerce Performance

Following are metrics from a large ecommerce site showing the correlation between the speed of the first page visited and propensity to convert on site.



Conversion Rate By Speed Of First Page

This graph demonstrates the clear link between page speed and the checkout rate. As page speed decreases for this ecommerce site, checkout rate increases.

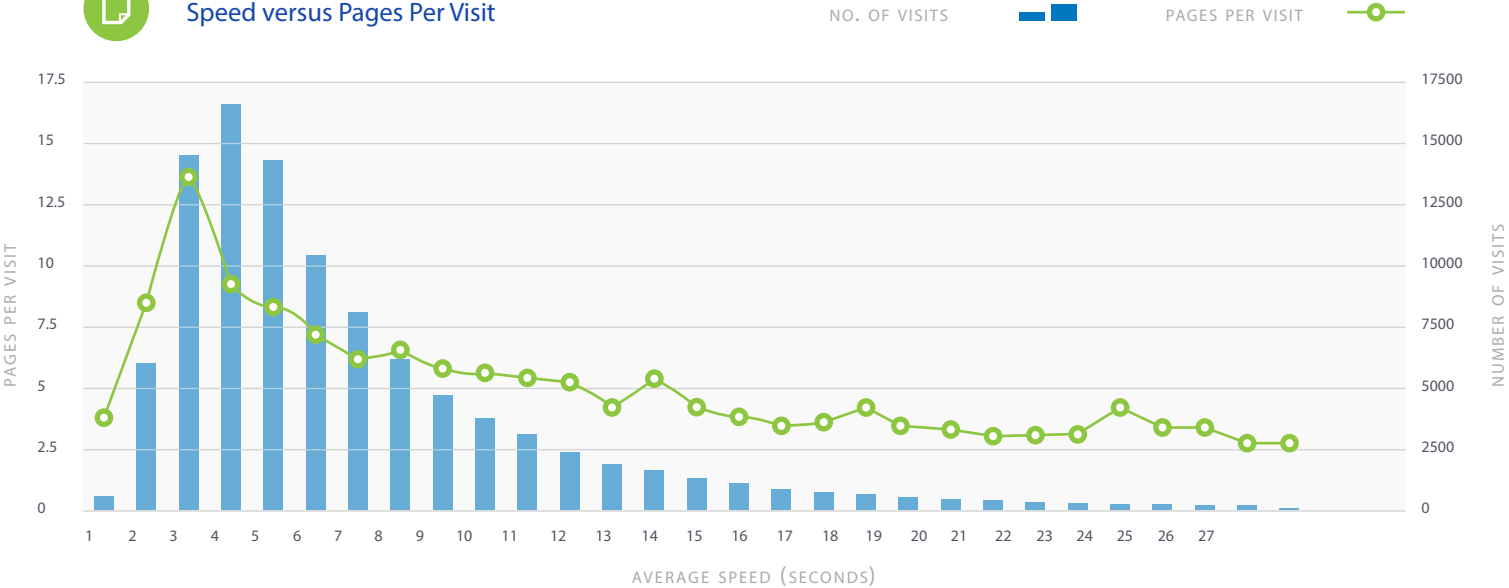


section.io also conducted an A/B test with Adore Beauty, a leading online beauty store that runs on Magento Enterprise, and found that customers who were served faster web pages:

- > Viewed 16.1% more pages
- > Viewed 9.4% product pages
- > Reached the checkout page 15.5% more
- > Saw an increase of 16.5% in the number of successful checkouts



Speed versus Pages Per Visit



Metrics To Measure Website Performance

There are a several metrics used to measure web performance (which we get into more in [Chapter 2](#)) but the main ones you should understand are:

Time to First Byte (TTFB) or HTML Load Time: When a user types in your web address ([www.yourMagentosite.com](#)) into their browser, there are a few steps that occur before the web page can even start loading. For example, if someone typed in [yourMagentosite.com](#) without the [www](#), it may first need to redirect to the [www](#) address. You may also have other redirects on your website, such as if you are sending mobile users to one site and desktop users to another. The browser needs to gather all of this information and connect to your website server (where your code is hosted) before anything else happens.

When the connection is established, your server starts sending the browser information in the form of an overarching HTML document that tells the browser what actions it needs to take and what files it needs to retrieve to build the page. This is where application code is executed and database calls are made, and if the HTML document is not cached (see [Chapter 4](#)), this process happens over and over.

An ideal TTFB is around 200 milliseconds which can be achieved when the HTML document is served from cache.

Start Render Time: Once the browser receives the HTML document, it starts building the page by making additional requests to your server, for everything from font and logo files to the text and images that make up that specific page. An average web page will need to make over [100 server requests](#) to gather all the content needed.

The Start Render Time is an important measure because it is when the viewer first sees the page appear in their browser. Although not all images and files may have loaded yet, this indicates to the user that the web page is loading.

Page Load Time: The page load time is probably the most common metric used to assess web speed, and it is the amount of time (measured in seconds) it takes from when the user first types your web address into their browser to when the web page is fully loaded in their browser.

Some browsers will indicate a page is done loading by including a progress bar underneath or to the side of where you type a website address. This measure is easy to understand which is why it's most often referenced. However, if you have a particularly large image or file that needs to load below the fold (so a user has to scroll down to see it), it would slow down your page load speed even if visually the user sees a complete web page.

The ideal page load time is under 2 seconds to prevent users from bouncing and increase engagement.

What Is Website Scalability?

In terms of Magento ecommerce sites, website scalability is the ability for a site to handle ever-increasing amounts of traffic and sudden bursts of traffic.

As a Magento site, you are likely to encounter an increase in visitors during a busy season, while you're running a sale, or due to an email marketing campaign. Having a website that doesn't scale well would mean all those potential customers encounter a slow website or can't connect to your site at all, at exactly the time when you're trying to take advantage of the increased traffic.

If you're investing time and money in bringing customers to your Magento site, you need to ensure the site stays up and fast so customers don't immediately leave the site or abandon their cart before completing a purchase.

The Time to First Byte measure indicates how long it takes for the browser to receive the HTML document from your server.

Scalability is linked closely to website performance, as a site could perform well at a small scale, but slow down for everyone if the number of visitors to a site increases drastically.

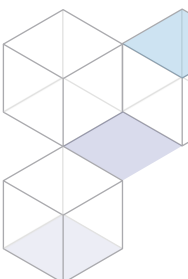


Improving Website Performance And Scalability

Luckily, website performance and scalability can both be addressed by the same solutions, which we will cover in the remainder of this guide. This is because ultimately both performance and scalability come down to the ability of your website servers to handle traffic quickly (performance) and with enough capacity that they continue performing well under an increase in traffic (scalability).

In [Chapter 2](#) we will show you how to measure your website performance and scalability, and then we will go over the aspects of your Magento website that need to be considered when optimizing for performance and scalability, including:

- › **Choosing the right hosting for your Magento site**
- › **How caching can improve both performance and scalability and how to utilize Varnish Cache with Magento**
- › **Content Delivery Networks and the benefits they bring**
- › **How to program your Magento 2 site for better performance and scalability**



2 | Data Driven

How To Measure Your Web Performance



Summary:

Measuring your current performance and scalability is important so you know what areas are most in need of optimization.

There are several ways to measure your website, including Real User Monitoring, Synthetic Testing, Load Testing, and Application Performance Monitoring.

To see how performance is impacting your user experience, you'll want to use a combination of back end and front end metrics.

In the last section, we went over why performance and scalability is important for your Magento website. Here, we will give you some tools to measure your website so you know where you're starting at in terms of page speed and more.

There are many tools that provide website metrics on everything from the number of visitors to your site to page load time, and it's important to understand;

- > What metrics are important and why
- > Which tools are best at providing you with accurate and reliable metrics

Performance Metrics To Examine On Your Website

Page load time: The time from the start of the initial navigation until the time the page is fully loaded in the web browser. This metric will be the longest since it includes all the steps to load your page, but it's important to look at the smaller metrics to understand where your page load is getting slowed down the most.

Redirect Time: The amount of time it takes to fully redirect to the correct domain (for example, if a user types in YourMagentoSite.com it will need to redirect to www.YourMagentoSite.com). If there are no redirects in place, this should be 0.

DNS Lookup: The time it takes for the browser to search for and find the IP address of your site.

HTTPS Negotiation: If your site serves content via HTTPS, meaning content is encrypted, the browser and server will need to exchange information to verify the SSL connection.

Server Connection and Server Response Times: Once the browser has located the IP address, the amount of time it takes to connect to your website, and the time it takes for your site's server to respond.

HTML Load Time: Otherwise known as the Time to First Byte (TTFB). The time in which the HTML document (the key to starting any page drawing in the browser) starts to be delivered to the web browser.

Start Render Time: The initial point in time in which the first non-white content (anything that is different from a blank page) becomes visible and is displayed on the web browser.

Document Complete or Document Content Loaded: When the HTML document has finished loading but other elements such as images referenced in the HTML document are still being delivered.

Fully Loaded Time: The time from the initial navigation until there are 2 seconds of no network activity after Document Complete. This will include any JavaScript activity that is triggered after the main page load.

To see how performance is impacting your user experience, you will also want to look at these marketing metrics:

Page views: The number of pages viewed by users on your site within the specified time period.

Pages/Session: The average number of pages viewed per user session. Repeat views of the same page are counted.

Bounce rate: The percentage of users who left your site after looking at just one page.

Session duration: Average time a user spends on your website. Keep in mind the user may not be actively engaging with your site this entire time.

Conversion rate: If you have set up goals correctly, the conversion rate will indicate the percentage of users who successfully completed a goal, such as made a purchase.

Ways To Measure The Performance And Scalability Of Your Magento Website

Real User Monitoring (RUM): As its name suggests, Real User Monitoring or RUM measures how actual visitors to your website experience it; what pages they view, how long each takes to load, when they exit or bounce from your site, and many more metrics about almost every part of their interaction with your site.

RUM records information about the visitor such as IP address, location, browser type, device type, and also what actions that user takes on your website.

The most commonly known example of a RUM tool is Google Analytics, a tool that almost every website marketer will be extremely familiar with. Unfortunately, out of the box, Google Analytics RUM is often misleading as it provides a small data sample for site speed;

Google states:

“Analytics restricts Site Speed collection hits for a single property to the greater of 1% of users or 10K hits per day in order to ensure an equitable distribution of system resources for this feature.”

In addition, Google Analytics only provides a mean site speed. When analyzing website performance, we should consider the median and measures within the 95th percentile as this reduces “noise” from outliers. These outliers are often outside the control of website developers and performance engineers. For example, they could be the result of customers on very poor equipment or connections.

Other RUM tools aimed at website developers dig even deeper into the experiences of real users on your website, providing more granular data and performance metrics that a marketer using Google Analytics for a general overview of site traffic and behavior probably would not deem necessary.

RUM is valuable because it shows how users are actually interacting with your sites, and could uncover issues that even the most stringent testing may not find. Because users are visiting your site from different locations, browsers, or devices, and have different ways of navigating around the site, viewing products, and checking out, RUM provides a wide range of analytics that will help demonstrate the average performance of your site for all types of users.

The one downside of RUM is that it relies on actual inbound traffic to your website. If you have just launched your Magento site, it may not yet be getting enough traffic to provide you with a good base of data on web performance or scalability. Without having much traffic to your website, another way to gather performance data even if you have a low number of web visitors is through **Synthetic Testing**, which we go into detail on below.

RUM is a method of passive monitoring, and is usually done by inserting a JavaScript snippet on your site.

Google Analytics provides the information mentioned above in a user-friendly interface so data can easily be monitored by non-technical employees.

Synthetic Web Testing: This type of testing, also known as active monitoring, is done by a web browser emulation that creates scripts to measure performance of a website as if they were an actual web user. This also allows for performance measurement before traffic is sent to a website or in a testing or staging environment, so that issues can be identified and fixed before potential customers would come across them.

Synthetic tests must be scripted to take certain paths through a website, so they are not necessarily a good indicator of how an actual user would navigate through a site. However, this type of measurement could be utilized to ensure the checkout process on your Magento site is running as expected after making a change, or to test page load time and see what elements of your website are performing well and where there is opportunity for optimization.

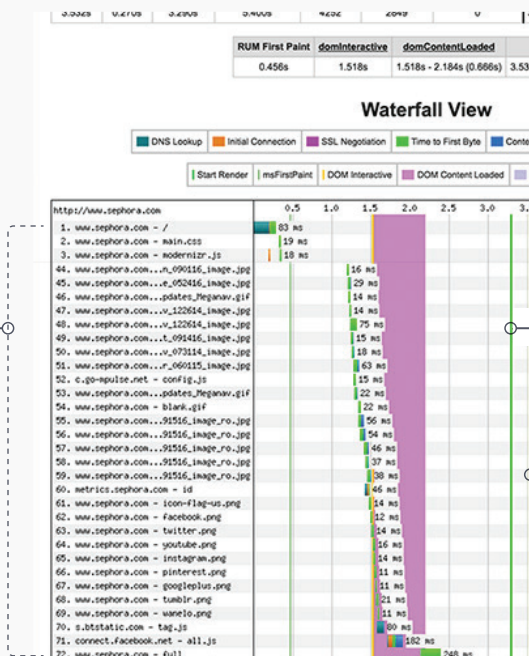
Synthetic testing can also help you see how your site will perform from different browsers and geographies even if you don't yet have users coming from those areas.

Synthetic testing is valuable if a website doesn't have enough real traffic to gather RUM data, and because it can be utilized by web developers at any time, 24/7.

One of our favorite tools to monitor the performance of your website is [WebPageTest.org](https://www.webpagetest.org), a free tool which allows you to run a synthetic test from a specified location, browser, and connection speed. By simply inserting the URL you're looking to test, WebPageTest will provide a waterfall view of your page speed with key metrics such as Time to First Byte, Start Render Time, and Page Load Time.

The waterfall view, as shown below for beauty store Sephora.com, demonstrates how long it is taking that particular browser type and location to get through each step in the process of loading a page, including the initial redirect and connection time, and then how long each specific element takes to load. This view would quickly show you, for example, if you had one JavaScript snippet slowing down your page load time, or if some of your images are taking much longer to download than others because they have not been sized appropriately.

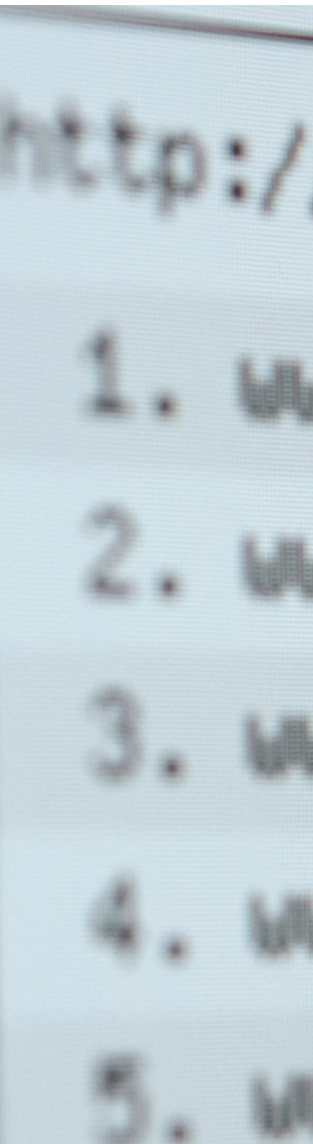
WebPageTest
www.sephora.com



These rows show all the elements that need to be loaded and how long each one takes to load

This page began to render in the browser after 3.29 seconds

This document was complete at 3.53 seconds, and visually complete at 5.4 seconds



You can also write scripts in WebPageTest to take certain paths through a website or do things such as login to an area of your site or go through a checkout flow. Here are some [instructions from WebPageTest on how to write these scripts](#).

For more information on how to utilize WebPageTest after you've implemented some of our suggestions for improved performance, see [Chapter 7](#).

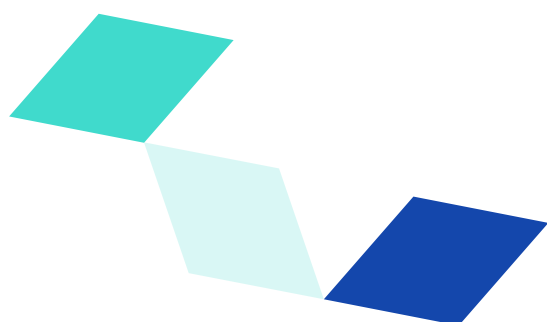
Application Performance Monitoring: This is a more technical type of monitoring which looks at back end specifics such as code execution, how many database calls are being made and how long they take to execute, and how your servers are being utilized. We recommend using an APM solution such as New Relic to monitor these metrics - find more information in [Chapter 7: Programming for Performance](#).

Load Testing: To test the scalability of your website, you could use a tool such as New Relic, which provides scalability metrics. To get a basic view of how your website will perform with additional traffic you can perform a load test. There are several online tools that perform load tests, which will mock up and send specified volumes of virtual users to your site at the same time. It is worth pointing out that Magento have included a load generation tool in their performance toolkit for Magento 2 if you wish to generate load to test your Magento store;

Load testing is intended to help you determine what amount of traffic your site can handle and identify weak points in your website architecture. Unfortunately, it is near impossible to simulate real user behavior. Variables such as user browser types, bots performing crawls at random times, bad actors like botnets, user locations, user network speeds, number of pages viewed, new versus return users, browser caches, actual checkouts or add to cart actions and more make for a myriad of options to consider when building scripts to generate and run a simulated load from a valid distribution of real browsers.

At best, load testing can quickly become a very expensive activity which will give you only an indication of load points in your Magento application. At worst, load testing can provide highly misleading results. In either scenario, your real load experience is guaranteed to be different from your load test results.

Once you've used tools to measure your current Magento site performance and scalability, you'll have a good baseline to improve upon. Take note of where your metrics are at now so you can see how each improvement changes them. In the next sections we will go through the impact of hosting, caching, and Content Delivery Networks on your website performance and scalability, and [Chapter 7](#) gets into the specifics about how you or your developer can implement these improvements for your site.



3 | The Host With The Most

Choosing The Right Hosting For You

Summary:

Hosting of your Magento application code, files and data is fundamental to the performance and availability of your website.

When choosing a hosting solution you will need to consider managed versus unmanaged offerings, and decide how many servers you need or if you can utilize a shared server.

You should also think about the security provided by your server solution, and the support given if server problems arise.

Hosting

One of the core decisions you will need to make when running your Magento website is which hosting provider to choose.

No matter what other services you add to your Magento site, hosting of your Magento application code, files and data is fundamental to the performance and availability of your website. If customers cannot reach your codebase running on the server, or it is executing slowly, you will lose customers and revenue.

The type of hosting provider you choose should be considered in the context of your complete solution design, including elements such as your caching strategy (see [Chapter 5](#) with respect to Varnish Cache) and your expected throughput, traffic profile and growth rate.


There are many hosting providers providing a range of services in the hosting space for Magento stores. Given the promises in the Hosting space of uptime, services, performance, support etc, it can be difficult to discern the right hosting partner for you. On the other hand, the likelihood is that you will find a number of partners who suit your needs in the market so you will need to weigh several factors when making a decision.

Hosting Options

Managed Hosting: A Managed Hosting offering means you won't have to think about patching, updates, server configuration and the details which go along with making sure the server is up and running at all times and that it is kept up to date for security purposes.

When contemplating a Managed Hosting offering for a Magento website, it is probably worth considering a Magento specialist hosting service rather than a generic managed hosting infrastructure. Given you are outsourcing a core component of your website performance and security with this option, you want to be sure that the provider can tune the environment appropriately to run Magento code and databases.

Traditionally, Magento hosting providers own or rent racks and servers in a data center. However, the growth and maturity of modern cloud hosting options have created extreme economies of scale and have driven many hosting providers to migrate from traditional hosting services to managed cloud services. This means that rather than owning or renting racks and servers, they resell the compute capacity from large cloud hosting providers alongside a layer of hosting management as a service.



Be sure that your hosting provider can tune the environment appropriately for Magento.

Self Managed Hosting: Conversely to the Managed Hosting option, Self Managed means you need to configure, patch and update the server/s yourself. While this brings great flexibility to the nature of the Magento solution you can run, it also comes with the overhead of running an activity which may not be your core strength. So while these options can present a headline cost which appears appealing in comparison to a Managed Hosting solution, the overhead cost of continually managing your server needs to be considered.

Server Options

Shared Server: Only an option for Managed Hosting, a shared server means you are effectively renting a part of a server from a hosting provider. A shared server is usually the cheapest Managed Hosting option because you will be sharing resources with a number of other customers of the hosting provider.

The downside of a shared server scenario is that your site can be severely impacted by other activity on that server. If another application hosted on the shared server is consuming all the resources on that server at a point in time, your website could become unresponsive until the hosting provider can scale the solution. Malicious or spam related activity by your neighbors on that server could also impact your Magento store's standing in global email and Internet filters.

Malicious or spam related activity by your neighbors on a shared server could also impact your Magento store's standing in global email and Internet filters.

Shared server options are usually the cheapest option and generally only viable for small Magento sites which are not growing.

Dedicated Server: Historically considered as a more expensive solution, particularly when it comes to scaling, a dedicated server means your website runs on its own server.

Previously this meant that you would have to understand exactly what capacity your website would require at its peak throughput moment and buy dedicated server capacity to serve that peak load moment. This process meant you would be sure to have sufficient server resource for that peak moment (contrary to a shared server situation), but your server resource would remain largely idle for the majority of the time - hence the cost effectiveness of this solution was questionable. A dedicated server hosting solution did not mean that your hosting solution was entirely isolated from other customers, as for example, you would still share networking and power resources.

Now, with cloud infrastructure solutions available, a dedicated server can be provided on shared infrastructure. Cloud providers dedicate a certain amount of resource for your application. Provided your application is built and deployed appropriately, you will be able to scale the application hosting footprint up and down. Because the cloud providers deliver resources from shared infrastructure and therefore can provide more effective costs, cloud dedicated servers can provide some of the cost benefits of a shared server with the performance and solution flexibility benefits of a dedicated server.

How Many Servers Do You Need?

Production: A production Magento store can run on one server. Many retailers have chosen to run Magento on one server to maintain simplicity of the deployment and reduce the cost of the server configuration or the cost of software running on those servers.

We recommend that all but the smallest Magento stores run in a High Availability (HA) hosting environment which would include two application servers with a load balancer in front. This setup will provide you with improved availability and scalability of the Magento website. With two servers in play, a failure of one of the application servers at any one time can be tolerated.

With two servers in play,
a failure of one of the
application servers at any
one time can be tolerated.

Recovery from server faults can also be handled more gracefully as servers can be pulled in and out of the load balancer while recovery, repair or restart is initiated. During this recovery time you will then have at least one healthy server to handle traffic. This setup is known as the “N+1” configuration for high availability. You have N servers to handle your regular workload, plus one extra that is always engaged so that you can lose a single server and still have enough capacity to service your customers.

We also recommend moving the Magento administration screens onto their own individual server. This will allow your operations team to perform all their content and order management functions on servers that are adjacent to the servers that handle the customer traffic. If a store administrator needs to perform an expensive operation like catalog reindexing, that work will take place on the administration server and will not consume valuable customer-facing resources.

It is important to consider your caching strategy when sizing server requirements. A well implemented caching strategy can massively reduce your server requirements. This is particularly the case as your customers generate load on your website during major events and promotions, as this is when the cache hit rates (and offload from your Magento servers) will be the greatest. Caching will also give you the best scaling multiplier available for these situations.

Staging: It is also prudent to run a staging server which has the same configuration as a production server. One staging server of somewhat lower specifications than the production server is usually sufficient to test impending production server configuration changes or code and database deploys.

Other Considerations

Security: If your website is accepting credit cards for payment online, and your infrastructure transmits those card details (regardless of whether your Magento application stores card details or not), you should be running on PCI compliant infrastructure. Hosting providers will be able to confirm immediately if they provide services on PCI compliant infrastructure.

Subject to the nature of the delivery layer running in front of your hosting service, you may also need to ensure your hosting provider can protect your website from attacks and malicious activity. We cover this in more detail in [Chapter 8](#).

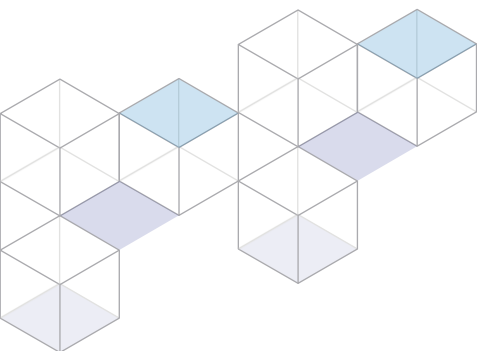
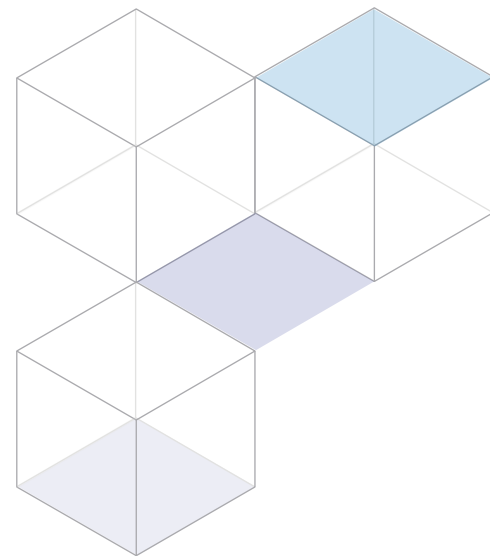
Support: The level and type of support you need may be subject to the type of hosting solution you choose (ie. Managed versus Self Managed). However, with any hosting support experience, you should look for demonstrated performance against the Service Levels which you agree with the Hosting partner. While some hosting partners advertise attractive support turnaround and capability, not all are able to deliver as advertised.

We believe it is also important to find a hosting partner with Magento specific capabilities, such as the required technical systems for High Availability Magento, like Redis/Memcache support, a shared file system, and MySQL. While Magento is “just a PHP application,” it does have some nuances which make supporting it easier when the provider has experience and capability with Magento specifically.

Magento Hosting Options

Following are just a few Magento hosting options for you to consider:

AWS
Cogeco Peer 1
ChinaNetCloud
Rackspace
Nexcess
Hostway
Anchor Hosting
Byet
Simple Helix
Zerolag
Tenzing



4 | Getting To Know You

An Introduction To Caching For Performance



Summary:

Caching is an easy way to improve the performance, scalability and security of your website. Magento strongly recommends using Varnish Cache as a caching solution.

Caching means storing copies of your web pages on additional local or globally distributed servers and in your users' browsers so that these pages can be served directly from these servers rather than from your Magento web servers.

Utilizing a cache effectively will dramatically improve site speed and the number of visitors you can serve simultaneously.

In this chapter we will review the fundamentals of web caching, which is one of the major ways to improve the performance and scalability of your Magento website.

Aside from leveraging data layer caching and browser caching, Magento strongly recommends the use of a caching tool, specifically [Varnish Cache](#), for the platform to run optimally. If you're looking for specifics on Varnish Cache and Magento, jump to [Chapter 5](#). If you'd like an overview of what caching is and why it's important for your website first, then read on.

Why Do I Need To Cache Content?

Before we get into what exactly caching is, you need to understand why caching is important. Ultimately, the main benefit of caching is serving faster web pages (performance) to significantly more users without your site slowing down or going offline completely (you guessed it, scalability). As discussed in [Chapter 1](#) of this guide, faster web pages lead to a better user experience, which means happier website visitors. Multiple studies have shown that users visit more pages on a website when it loads faster. Improved performance and higher conversion rates also mean search engines view your website more favorably, which improves your SEO value and means more people find your site.



FASTER WEB PAGES

>



BETTER USER EXPERIENCE

>



HAPPY CUSTOMERS

=



MORE REVENUE

Every request served from inside a browser's cache or from your delivery infrastructure cache (such as Varnish Cache) is one less request made to your web servers; one less request your servers need to connect with, compute and send. These requests are faster for the browser and reduce the stress on your web servers. Caching can keep your pages loading quickly at traffic levels 10 to 100 times greater than might otherwise be the case.

What Is Caching?

Now that we know why caching is important, let's take a look at what it actually means and how caching makes your website faster and more scalable.

Every time a user visits a web page, they are using a web browser to request and assemble that page from the website's server. The server holds all the application code and files needed to assemble that web page, including the HTML document (instructions to build the rest of the page), the images, text, styling, and more. On average, a browser makes upwards of [100 requests](#) back and forth from the website's server to build a complete webpage.

Without any type of caching, whenever a user visits that page they make those requests all over again, and every other person visiting that web page is making the same requests. If there are lots of people accessing a page at one time, the server slows down and takes longer to deliver the web page to everyone. **Slow web pages = unhappy visitors.**

Caching solves this problem by storing a copy of the assembled web page and components in different locations. These copies are temporarily stored somewhere other than the website server, so the browser doesn't need to go all the way back to the server each time a visitor loads the same page.

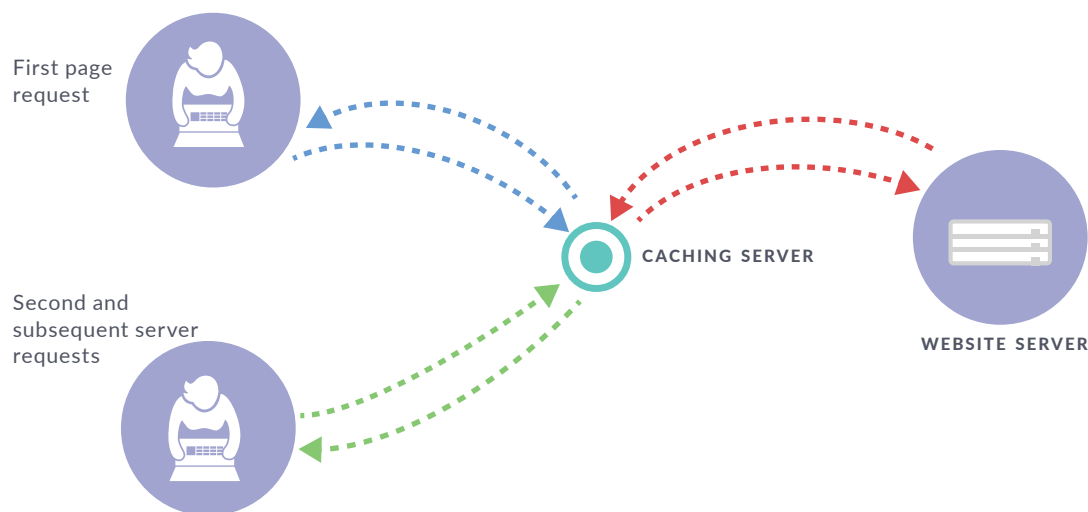
You may be familiar with the phrase “clear the cache” or “clear browser data” - this is one of the first things engineers ask you to do when troubleshooting why a web page isn't showing up correctly. Clearing the cache deletes the files that have been saved in a browser or server cache, forcing the browser to go back to the website server and download a “clean” copy of the web page. Here are two major caching locations and how they work.

Browser Caching: One way to cache content is to do it directly on the hard disk of a user's device. When the web server is set up properly, web browsers do this automatically for web pages, so they don't need to go back to the website server to download every single element again. For example, a website logo is often repeated on each web page a visitor goes to. If that logo is in the browser cache, the browser doesn't have to re-download it if the user visits the same pages in the future.

Server Caching: Server caching means web pages are cached closer to the website server, rather than on a user device. If you install a cache on your website server, it means you are keeping copies of the relevant files and instructions in that cache.

The first visitor to a website after a cache has been installed or cleared will be directed to the web server, which will then send a copy of the pages to both the cache and the end-user. The next time someone requests the same pages, the cache will be able to directly fulfill their request, resulting in shorter page load times.

Websites can control how often their cached content needs to be updated: the cache will re-collect a new version of the web page from the server for the first visitor to a web page after the cached content has expired, and then deliver that content to that visitor and subsequent visitors until that newer copy also expires. This graphic illustrates how it all works:



Cache Hit vs Cache Miss: To get into the terminology of all this, a “cache hit” means that the cache successfully delivered your visitor cached content, whereas a “cache miss” means the request checked the cache, but the cache didn't have the relevant content stored. A “cache pass” gets the content directly from the server without checking the cache first. The higher the cache hit percentage, the more often people are getting content delivered to them through the cache, meaning web pages are loading faster for them and there are less requests going to your website server, which also decreases your server hosting costs.

What Web Content Is Usually Cached?

There are some types of files that are frequently cached by websites, some files that can be cached but that many websites do not cache because they are seen as “risky” (we’ll get into that later), and others that are never cached.

Files that are **frequently cached** are ones that are the same for all users and don’t change often. They may include:

> **Static images**

- Logos and brand assets
- Product images

> **Stylesheets** (the code that dictates the font, colors, etc. used throughout the website)

> **Javascript files** that don’t change (for example, the the Javascript framework you use, like JQuery)

> **Downloadable files** or other content

- PDF product documents
- Video files of product demonstrations

Files that **can be cached but rarely are** include:

> **Full HTML pages**

> **Partial HTML pages**

> **HTML Snippets**

> **AJAX Snippets**

> **Javascript files or other code** that changes more frequently

Files that **should not be cached** include:

> **User-specific data** such as account information that is different for each visitor

> **Any sensitive data**

- Banking or credit card information
- Security tokens

Caching HTML Documents: The HTML document is the first piece of information that a web browser receives when it loads a web page. This document includes all the information needed to instruct the browser to load the elements of a page, including stylesheets, logos, images, header and footer files, and more.

The process to generate a web page’s HTML document is where most of your website server’s resources are spent. However, most caching solutions and Content Delivery Networks focus on caching static files, and do not automatically cache full HTML documents.

The full HTML document is critical to the behavior of a web page. If it is cached incorrectly it could result in a page whose layout appears completely off, or one that displays the wrong user account data.

Despite the considerable speed and resource-freeing benefits for your web servers, it can be risky for websites to cache their HTML documents if your development and operations teams aren’t able to properly test their caching configuration and ensure everything will work as expected on the live website.

Most CDNs do not allow for flexible configurations and do not have a testing environment. This is the main reason that most sites still direct users back to their servers for the HTML document. To cache an HTML doc, developers must have the ability to implement flexible configurations within their caching and/or CDN solution, and also to test these configurations before they go live. [section.io](#) provides these benefits, which you can read more about in our [HTML caching whitepaper](#).

Do I Need A CDN?

Many people assume that the main purpose of Content Delivery Networks is to store and deliver static cached content from server locations across the globe. However, modern CDNs can do much more than cache static objects including protect against DDoS attacks, block harmful bots, and cache dynamic content including HTML documents.

While you do not actually need a globally distributed server network (ie. CDN) to take advantage of some of the benefits of caching, utilizing a CDN to cache and serve objects can make the process relatively simple and give you the best chance of increasing the performance, scalability and security of your website.

Using a CDN means you will have the added benefit of distributed servers to deliver your content to worldwide users, additional elastic capacity, and increased security and protection from attacks. Using a CDN built specifically for Magento also means you will have all the tools and technologies your developers need to manage and maximize the cache performance of your website.

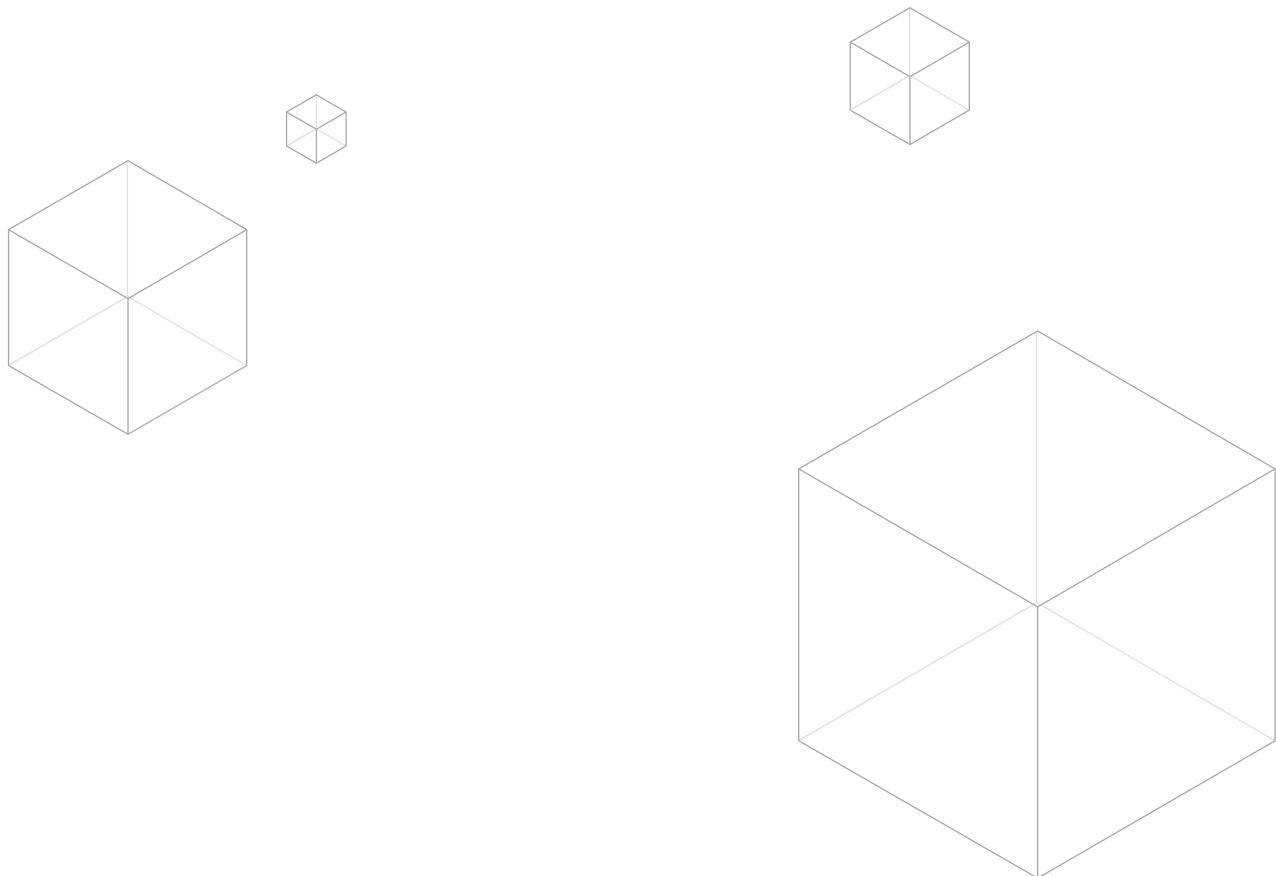
In [Chapter 6](#) we go into more detail on the benefits of utilizing a CDN for your Magento website.

What Tools Are Out There For Me To Cache My Magento Website?

There are a number of caching technologies available, including Squid and Nginx as open source options. For Magento, the caching solution that should be utilized is Varnish Cache which is an open-source HTTP acceleration proxy that can either be installed locally (such as on an additional website server or in a different part of your main website server) or on a globally distributed server network.

Magento is built to work with Varnish Cache, and Magento strongly suggest utilizing a Varnish Cache configuration to increase the performance and scalability of your ecommerce store, especially for Magento 2.

For more information on setting up Varnish Cache for your site, read [Chapter 5](#), which is all about Varnish Cache and Magento.



5 | Digging Deeper

How Varnish Cache And Magento Work Together



Summary:

Correctly configuring Varnish Cache is the most important thing you can do to immediately improve your performance and scalability.

Varnish Cache is a reverse proxy which sits in front of your web server so that customers reach cached content before they go back to your server.

To get the best performance out of Varnish Cache, ensure you are managing cookies and caching partially dynamic pages so as much HTML is cached as possible.

No discussion on the optimization for performance and scalability of Magento would be complete without consideration of Varnish Cache.

As an advanced ecommerce Content Management System (CMS), Magento can be a resource hungry application, consuming the Central Processing Unit (CPU) to execute the PHP code upon which the application is built. Without a caching strategy, the more users that visit your website, the more PHP will be executed from your servers and the more CPU those servers will require.

One strategy to deal with this repetitive execution and consumption of CPU has been to “throw more hardware” at the problem. At first glance, it would seem to make sense that the more CPU consumption demanded, the more CPU we should provide.

However, when the website activities are analyzed, we generally find that many of the activities on a website by users are repetitive in nature both for any one user and also across multiple users.

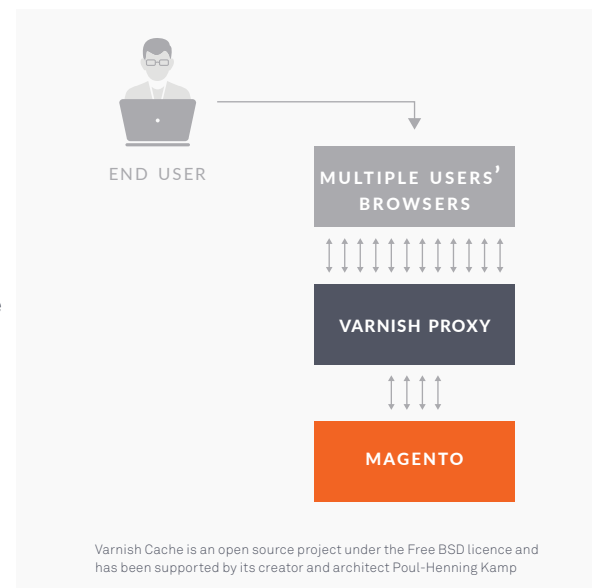
For example, a single user may view the home page for your website multiple times within a session and then multiple users will also request that home page for each of their sessions. As the owner of a Magento website you can either provide enough CPU to generate and serve that homepage multiple times to each and every user, or, because the users are all requesting the same thing, you could generate it once for the first user, keep a copy of it somewhere and then serve the next and following users the copy of the page. This makes the page both cheaper to serve (in terms of CPU) and faster to serve (as there is no time spent producing the page). This caching approach is where Varnish Cache comes in and where it shines when it comes to Magento websites.

Caching saves you money by reducing server costs, and speeds up your site at the same time.

What Is Varnish Cache

Varnish Cache is a server you run between your customers and your Magento web server. The Varnish Cache server acts as a proxy for the web server, and will receive and either serve inbound requests directly from its cache, or pass the request back to the origin server to be dealt with if the cache does not possess the required assets. To your customer, their browser sees Varnish Cache as your web server, and thus Varnish Cache acts as a shield for Magento.

Varnish Cache is exceptional in this capacity as it is extremely fast, capable of scalability, and is highly configurable thanks to the programmatic approach to setting the caching and handling configurations. Technically, Varnish Cache is a HTTP accelerator which is run as a reverse proxy. It can be referred to as a reverse proxy server although there are a few technical elements in Varnish Cache which cause some purists to claim it is not a reverse proxy server.



Varnish Cache Versions

As of October 2016, Varnish Cache is currently running Version 5. If you are using Magento 2.x for your store, you will want to run Version 4 or newer.

Earlier versions of Magento work well with Varnish Cache 3 but may require an additional plugin to maximize the benefits of Varnish for your website. Plugins such as Turpentine can be helpful to optimize your Magento website for use with Varnish Cache. Your best option today is to run Varnish Cache 5, and work towards maintaining your Varnish Cache installation as newer software is released.

Setting Up Varnish Cache For Magento

On Premise: Varnish Cache can be deployed in a single or multiple server configuration “behind your firewall” in your hosting infrastructure. You will need to run one or more Varnish Cache servers. We would recommend that for High Availability, the Varnish installation includes no fewer than 2 servers unless your Magento website is very small (say, less than 10,000 pages served per month).

Subject to the scale and growth of traffic on your website, you should also give consideration to management of the scalability of the Varnish Cache servers to ensure your Magento website can handle whatever volume of traffic you point at it.

By design, Varnish Cache does not handle HTTPS traffic so when installed you should ensure that Varnish Cache has a layer running in front (such as your load balancer) which has your website’s SSL certificate installed to handle SSL handshakes with the client’s browser. Typically, a load balancer would be configured with your HTTPS certificates and then convert the traffic to HTTP so that Varnish Cache can do its job.

For diagnostic and optimization purposes you will need to make sure you set up the servers in such a way that you can access the Varnish Cache logs and traces. See for example, this [Varnish Cache guide on logs and metrics for Varnish Cache 4.1](#). These sources of information will be important to help you understand how successful your Varnish Cache configuration is. They will indicate how much of your website is being served directly from the Varnish Cache layer, and where you should work to improve the Varnish Cache server configuration and hence the performance and scalability of your Magento website.

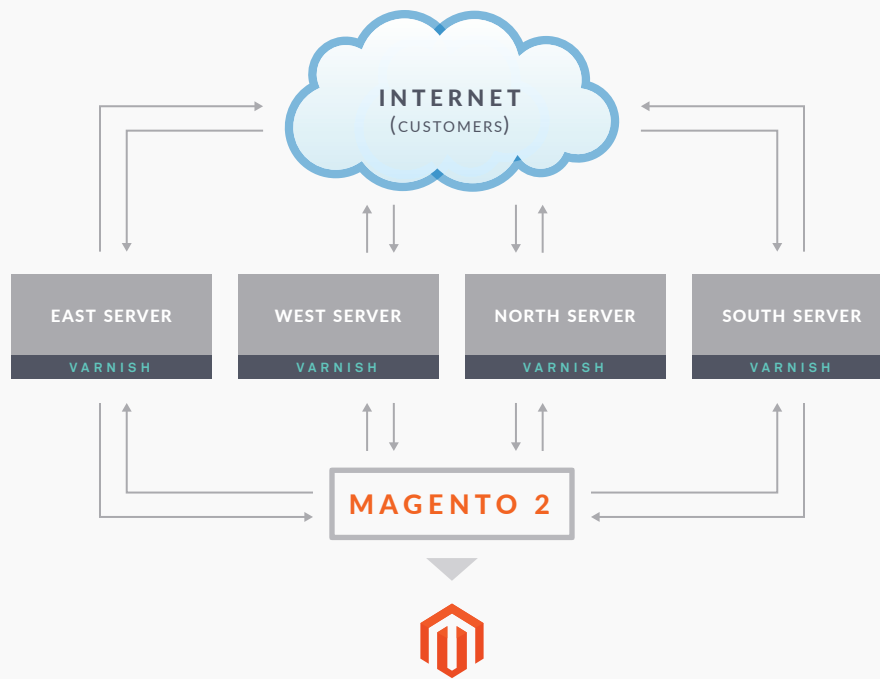
Varnish Cache does not handle HTTPS traffic so when installed you should ensure that Varnish Cache has a layer running in front which has your website’s SSL certificate installed.

Because you need to keep mission-critical websites running all the time, you need a high availability setup. When you run multiple servers it becomes increasingly important to put all the Varnish Cache logs into a single tool for analysis. You don’t want to be jumping from server to server to try and find a failed request. Aggregating your logs into a single tool is very useful and cuts down debugging time and reduces errors.

For development purposes, your developers should have access to a copy of your production Varnish Cache servers in their development environment. Your website developers should be testing the compatibility of the components they are developing with Varnish Cache to ensure that Varnish Cache can effectively cache as much as possible of the component, and that the component and the current Varnish Cache configuration do not conflict and cause website errors or outages.

Distributed Varnish: Instead of setting up Varnish Cache inside your hosting you could install Varnish Cache on servers that are located closer to your users. In this setup you still have a single installation of Magento at your hosting company but Varnish is not installed there - it is installed in multiple datacenters around the world. This is essentially the core of a Content Delivery Network, but this setup handles more than a normal CDN. It also handles the caching of complete web pages, so it fits the popular term “Full Page Cache”.

A limited number of providers can deliver a globally distributed Varnish Cache platform. Such a platform can provide you with a click-and-go Varnish Cache layer for your Magento website that includes all the Varnish Cache setup requirements of high availability, elastic scalability, metrics, logs and alerting and HTTPS management. Some providers, such as section.io, may also deliver an out of the box development environment.



By pushing your Varnish Cache layer out to a distributed platform, your website performance will be further enhanced by delivering your website content directly from Varnish Cache servers which are closer to your end users. Pages load faster when the content your users need is closer to them because the data doesn't need to travel as far.

This type of Varnish Cache setup for your Magento website can deliver all the benefits of Varnish Cache for Magento and a global Content Delivery Network rolled into one. To learn more about CDNs and consider if this is the right option for you, read [Chapter 6](#).

Optimizing With Varnish Cache

A more detailed discussion of how to optimize Magento 2 with Varnish Cache is addressed in [Chapter 7](#). However, at a high level, the goal of optimizing your website for performance and scalability with Varnish Cache is to have Varnish Cache serve as much of your website as possible. This includes images, static files (CSS, JavaScript, etc.) and the actual HTML document itself. This is known as Full Page Caching.


By configuring Varnish Cache and your website so that Varnish Cache serves as much content as possible, you are reducing the work your web servers have to do to generate that content. This will both speed up your website and increase the amount of traffic your website can handle at any time.

Which Content Can Varnish Cache Serve For Your Webpages?

Every web page is comprised of an HTML document, static objects, and requests to third parties which may be inserted in the form of JavaScript snippets, such as those that pull information from social media sites or track visitor movements. Commonly, these objects served are also broken into “static” and “dynamic” elements.

Caching Static Objects: Static objects would include items such as images, the Cascading Style Sheets (CSS), and the JavaScript that is required to build up the page. These items are referred to as static as they do not change from one user to another. If a user requests a Magento product page, then generally, then next user will see exactly the same product image as the first user regardless of which pages they have visited previously on the website, how many items they have in their shopping cart, and whether the stock is running low or the price has changed.

Varnish Cache does an excellent job of caching and serving static objects and it should be the goal of every Magento website to ensure as much static content as possible is served from a caching layer. Ideally, that caching layer is running closer to the end user (such as in the instance of a Distributed Varnish Caching layer discussed above), so that the objects can be served faster into the end users' browsers.



The more HTML you can
serve from Varnish
Cache the faster and
more scalable your
Magento website will be.

Caching Static Objects: The HTML document of webpages is generally the first resource your Magento web application will send to a user's browser after they have requested a page from your Magento store. This is the set of instructions and source of data with which a browser can build a page. It will include:

- Instructions for the browser as to which images to fetch, which styles to use, what text to draw on the page and in what order;
- The data that makes up the page content such as product description or price;
- Whether the user is logged in and what their user name is in addition to the number and value of items which may be in their cart

Some of the above elements can be considered dynamic in that they may change for every user based on that user's actions on the website. For example, the contents of a user's cart will be unique to each user.

Where pages are considered dynamic, generally we would say that they are not cachable as you don't want to serve stale or incorrect content to the next user. However, due to the programmability of Varnish Cache you can improve the cacheability of these "dynamic" pages.

The majority of PHP execution time on a Magento server is consumed with generating and serving HTML, so the more HTML we can serve from Varnish Cache the faster and more scalable the Magento website will be. In fact, we see that approximately 50% of the CPU cost of Magento is generating category pages, and 25% is generating product pages. Since these pages are very sharable between users it makes sense to generate them a single time and share them between users.

Below are three high level areas to address to make sure as much HTML as possible is served from your Varnish Cache layer.

1. Managing Cookies and Sessions

Often, Magento sets cookies on a user early in that user's session which can force Varnish and Magento itself to identify that user's requests as unique, when in fact they are still requesting the same content as other users. By managing cookies in Varnish Cache you can share HTML between users.

2. Recognizing Partially Dynamic Pages

Magento product pages are often considered uncacheable for all users as you may make pricing changes to the product, or perhaps the product availability could change throughout the day. If you run out of stock, you would want that state to show as quickly as possible.

While the HTML for these pages is dynamic by virtue of the potential changes to the HTML from stockout or price change, it does not render these pages uncacheable. Using Varnish Cache settings you can manage explicitly for how long the page can be cached in Varnish Cache before a new object is fetched. If a popular product page is being requested by 1000 users a minute, you could ask Varnish Cache to keep a copy of the page for a max of 15 seconds which would reduce the requests for the page to 4 per minute from a potential 1000. This ensures the page is regenerated every 15 seconds while massively reducing the page load speed and cost to run the end servers. In Magento 2, any updates to product pricing automatically send cache purge requests rather than waiting until the set cache time expires.

3. Isolating Dynamic Components in a Page

Personalized content in pages such as the number of items in a user's cart is more difficult to deal with. However, you can isolate the dynamic content to certain parts of the page and serve the remaining page content from cache by using techniques such as Edge Side Includes or AJAX that fetch and render the dynamic parts of the page. Magento has chosen to favour AJAX calls for personalization in Magento 2.



6 | The Value Of Network

Considering The Benefits Of Content Delivery Networks



Summary:

CDN technology can significantly improve the performance, scalability and security of your Magento Website.

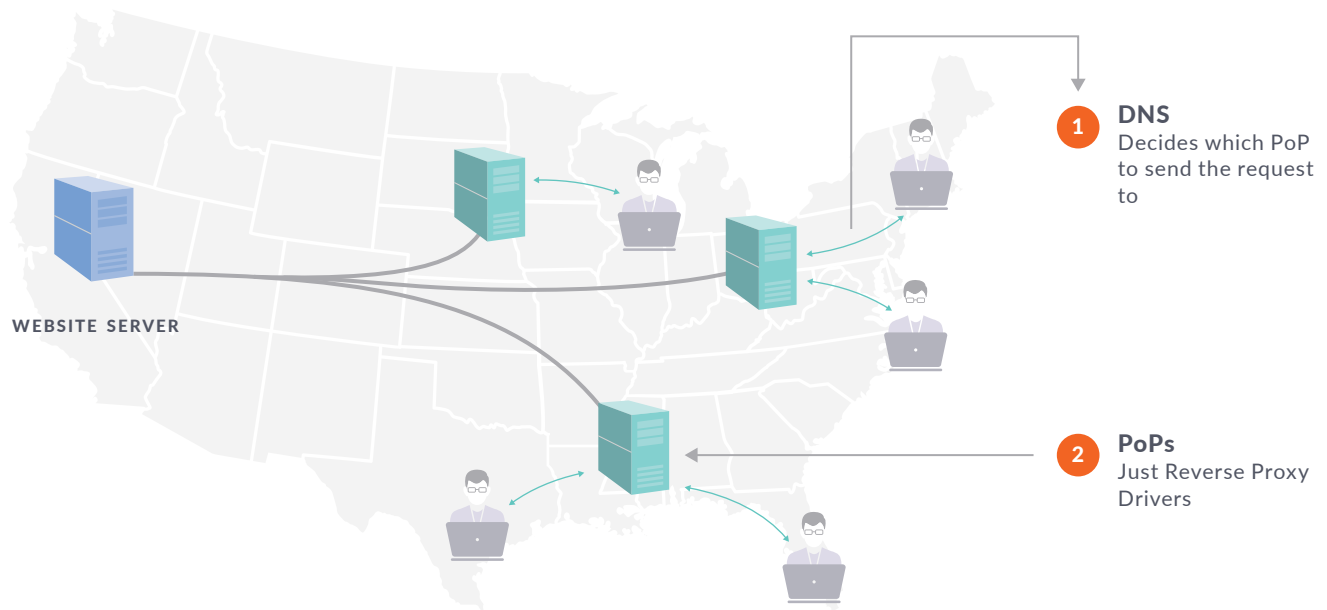
The CDN you choose should specifically fit your Magento application.

Your team should be able to use the CDN safely and effectively and your developers should be comfortable with configuring and testing your CDN solution.

What Is A Content Delivery Network?

CDNs are networks of servers distributed around the Internet. They are traditionally fixed networks of servers. Those servers are physically placed in locations called Points of Presence or “PoPs” that are designed to be closer to your end users around the globe.

CDNs’ PoPs act as a middleman for your web server and will intercept the end users’ requests. By intercepting the request and acting as a proxy, the CDN PoP has an opportunity to serve the request directly, block the request if it is malicious, or in some other way improve the response to the user (and hence the user experience and the security of the website).



CDNs Consist of Two Elements:

1. A layer of DNS technology

The DNS layer decides which CDN PoP to send any user request. The user request will be served by the CDN PoP closest to them, so a user in Europe would get directed to a different PoP than a user in the US.

2. Reverse Proxy Software

The PoPs are servers running reverse proxy software. A reverse proxy acts on behalf of your web server, by taking external requests (ie. from your customers) and determining how they should be handled. It is the reverse proxy software running on any CDN PoP which adjusts, blocks or handles the user traffic, and thus this software is the core of the CDN. Many CDNs wrap this technology in a “black box,” and do not give you visibility into the nature of the reverse proxy software running on their CDN or how it has been modified.

Examples of reverse proxy software found in CDNs include:

Varnish Cache: An open source HTTP accelerator which caches content, including the whole HTML document. Due to its programmability, Varnish Cache can be customized to perform a wide variety of functions in addition to caching.

Nginx: An open source server which can be run as a reverse proxy. Functions including caching and content rewriting (ie. running the programming language LUA or web application firewall).

ModSecurity: An open source Web Application Firewall which can detect and/or block a range of requests by parsing the request to look for a match to certain content (such as SQL injection attacks within HTTP requests).

Javascript Detection Bot Blocking (or Anti Scraping) Proxies: These reverse proxies query and detect for a browser’s ability to execute Javascript before allowing that browser to send its request to the origin. This has the effect of blocking undesirable bots from scraping your web content or causing a Distributed Denial of Service (DDoS) Attack.

Front-end Optimization such as the PageSpeed Module: Built and maintained by Google, this open source reverse proxy can perform a very wide range of front-end optimization activities to a webpage including resizing images, rewriting image types, engaging lazy loading of images so that images below the fold on long webpages aren’t immediately loaded, and prioritizing critical CSS.

Edge Rewriting Proxies: As noted above, Nginx running LUA can be used as an effective content rewriting proxy that enables developers to compile content for the web page at the “edge” of the delivery network (ie. a CDN PoP), or change the nature of the content based on information provided to the edge by the browser. This is useful for aspects such as JavaScript tag injection.

Why Choose A CDN For Your Magento Website?

To provide a framework around how to choose a CDN, first, let’s consider the core reasons you would choose a CDN to serve a website in the first place.

Scalability: Using a CDN can help offload requests and compute activity from your website servers. This means significantly greater scale as your server isn’t handling every single user request.

Performance: Using a CDN can bring content closer (in network time) to your users so that round trips required to deliver a page are faster.

Security: Using a CDN can block false or harmful requests from reaching your web servers and potentially stealing private data or bringing your site offline.

There are a number of other secondary reasons why CDNs can be attractive for Magento websites but usually the above are the main reasons bringing a CDN into your website delivery infrastructure.

In order to choose the right CDN for your application you’ll want to consider what goals you are trying to accomplish as well as the needs of your development team and users.

Choosing A CDN Can Be Confusing

There are over 40 global CDNs available today and there are quite a variety of options available within each one. If you go shopping for a CDN for your website, you may hear some of the following marketing statements:

Vendor 1: *We have the best CDN as we have Super PoPs with the best peering relationships to the telecommunications companies. Our PoPs are on the Internet backbone so content is served faster.*

Vendor 2: *We have the best CDN as we have the most PoPs worldwide. Our PoPs are closer to your users so content is served faster.*

Vendor 4: *We have the best CDN as our PoPs are the newest. Our infrastructure uses Solid State Drives so content is served faster.*

Vendor 4: *We have the best CDN as we have superior networking and shared cache between our PoPs. Content moves faster and less often to and from your origin so it is served faster.*

Vendor 5: *We are the cheapest CDN.*

One very important consideration which is often ignored is;

- › Which is the CDN your developers can actually use effectively?
- › Can your developers safely and effectively drive the CDN to deliver the promised performance, security and availability?

It can be challenging to work through the truthfulness of the CDN marketing statements and the extent to which any one of those statements may be important to your team or most importantly, your customers.

Choosing A CDN Which Your Website Engineers Can Use

This is the most critical element of choosing a CDN but is often overlooked.

If your development team cannot effectively use the CDN you choose, they will not be able to deliver the promised benefits. Many CDN purchases are plagued by implementation and update issues to such an extent that businesses are left with CDN contracts over long periods even when the CDN is not providing the caching or security benefits promised at the start.

To effectively drive maximum benefit out of a CDN, Magento Developers and Operations Engineers need:

Workflow integration: Most CDNs only run in production which requires developers to break their continuous integration workflow and can lead to problems which only appear once a site or application is live.

Consistent and Real Time Diagnostics: Teams should have access to immediate data and metrics to help them diagnose errors and find improvement opportunities. The metrics and measurement platforms should be consistent between their development, test, staging and production environments.





Choose The Best CDN For Your Magento Website

When deciding which CDN which will work for your Magento website it is worth stepping into the market with a shopping list. Without the right set of features to work with your website, you could be left with a functionally broken website, inadequate offload to the CDN, and/or a major drain of scarce technical resource being applied to implementation and management of the CDN. Some CDNs can hit just a few of the general requirements while others will hit many and become so complicated your tech team can't manage the options with the available resources.

Some of the items on your shopping list should include;

Ability to Cache HTML with Varnish Cache: Magento 2 is designed to work with a full page caching layer in front of the web servers. And the best full page caching layer for Magento 2 is Varnish Cache.

As Magento have noted:

“We strongly recommend you use Varnish in production. The built-in full-page caching (to either the file system or database) is much slower than Varnish, and Varnish is designed to accelerate HTTP traffic.”

Your CDN should be able to cache your Magento website HTML and ideally be running Varnish Cache in the CDN as the caching reverse proxy.

HTTP/2 Support: Modern web browsers support HTTP/2 which provides a 10-20% page performance improvement over HTTP1.

HTTP/1.1. You should not use a CDN which requires you to move static objects onto a separate domain as this is now a website performance antipattern.

Push or Pull CDN: Do you have to change your application or workflow to push content up to the CDN or will the CDN pull the content from your origin in an automated fashion?

Cache Clear: Can you clear the CDN cache easily if there are bad files cached at the CDN? How long does the cache clear take?

Cache Control Headers: Will the CDN observe your Cache Control settings?

SSL Encryption: Does the CDN support termination of SSL traffic at the CDN? Can you bring an Extended Validation Certificate to the CDN? What are the additional costs of each?

POST Support: Can your CDN support GET and POST requests? – the latter being relevant to many forms on websites.

Customer Support: Can you reach out to the CDN customer support easily? What is the response time and quality of initial response likely to be?

Change Required: How much do you need to change your application or core hosting infrastructure in order to support going live with the chosen CDN?

CDN Configuration and Management: How much ongoing configuration and management of the CDN will you be left with and does your team have the skills and time to do this?

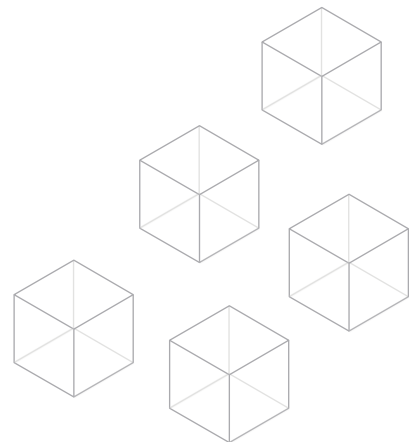
Choosing The Best CDN For Your Customers

Provided you can choose a CDN which works well for your website and development team, the next step is to consider which one will work best for your customers.

The best CDN for customers is going to be the one which provides the best speed boost for your content. How can you predict the CDN which will deliver the fastest content for your users?

Some CDNs will have Points of Presence (PoPs) geographically closer to your users and some will have PoPs which are along the Internet backbone and closer in terms of network distance. Other CDNs may have newer and more responsive infrastructure.

In reality, all CDNs perform at various levels at various times for various users subject to the performance of the network and infrastructure in the delivery chain for that user at that point in time. The milliseconds differential in network speed pales compared with the increased performance from having more items stored in and served from the CDN. This is entirely dependant upon the ability of the development and operations team to effectively use the CDN, and on choosing a CDN with the right tools (such as Varnish Cache) to support your Magento website (as noted above).



Choosing The Best CDN For Your Wallet

CDNs have a variety of ways of charging for services which may include;

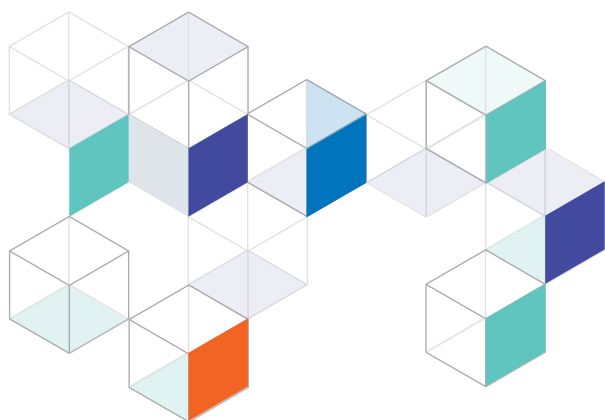
- > Traffic served
- > Requests served
- > Peak bandwidth utilization
- > Edge Storage
- > SSL traffic premiums
- > Extended Validation certificate premiums
- > Set up charges
- > Professional service fees for modifications
- > Overage penalties

And more...

We have often seen commitments made to CDNs on the basis of say, a traffic fee, only to later find out that the SSL certificates and SSL traffic were not included in the pricing (which in fact can be much more than the quoted HTTP traffic costs). It is not appropriate or required in today's technology landscape to pay for SSL certificates to be deployed on a CDN.

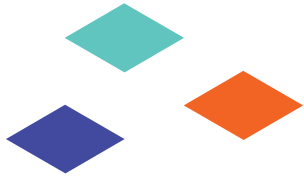
We recommend you choose a CDN with simple and transparent pricing which relates directly to the success of your ecommerce business. Web engineers and developers should not be penalized financially for activities such as the frequency with which they clear the cache.

We also recommend you think about what pricing will work for you as your Magento website grows and attracts more visitors: if a CDN has an attractive rate up to a certain amount of traffic, but jumps if your traffic increases (which is what you want!), it may not be best for you in the long term.



7 | The Meat Of It

Programming for Performance On Magento 2



Summary:

There are several things you will need to take into consideration when programming your Magento 2 website for performance and scalability, including the codebase, database architecture, and extensions you use.

Below we give your development team options for enhancements and guidance on where to install Varnish Cache and how best to configure it.

This section is intended to give your technical team some solid recommendations on how to implement Varnish Cache and improve the performance and scalability of your Magento 2 website. If you aren't the person in charge of making back end changes to your Magento configuration, we recommend handing this chapter to them. Now on with the show!

Updates In Magento 2

Magento 2 has made significant steps forward in terms of codebase, database architecture, extension quality and application design for caching.

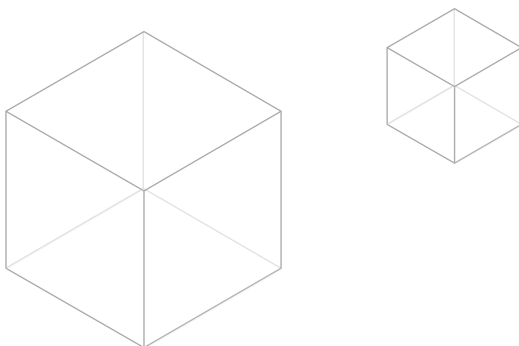
Some of the key improvements Magento 2 delivers are:

Codebase: This has been totally refactored with a focus on performance. The codebase also has significant unit testing coverage which allows code changes to be validated by automated tests as part of the build/release cycle. This improves code quality on an ongoing basis.

Database Architecture: Magento has retained the EAV model present in Magento 1 however there have been some optimizations, such as the ability to have 3 separate master databases in Enterprise edition: Checkout, orders, and product data can each use a separate master database.

Extension quality: The world of extensions in Magento 1 was an area fraught with danger as there are many poorly built extensions that work at low traffic volumes or with a small number of product items but then have significant performance issues once traffic or product counts rise. The Magento 2 strategy has been to throw away all existing extensions and define a new process to build and submit a Magento 2 extension that involves rigorous code quality checks before an extension is allowed to be offered in the [Magento marketplace](#).

Application design for caching: Magento 2 is designed to work out of the box with the Varnish Cache which is a lightning fast HTTP acceleration tool. Content cached and served from Varnish Cache is the fastest you can send.



Programming For Performance

The total time spent loading a webpage is a combination of the time it takes for the browser to request and receive the HTML document (back-end time) and the time it takes for the browser to download all of the resources (such as images, JavaScript and CSS) in the HTML document and complete processing of these resources in the browser (front-end time).

Back-end Performance

Improving back-end performance in Magento 2 has several areas of focus:

Application code: This is the code needed to perform functions like generating HTML documents or adding an item to a user's cart. There are several areas of code that need to be considered:

> **Magento core code**

This is the core Magento code base that provides all of the out of the box functionality for Magento.

> **Custom application code**

This is the code that you (or your development team) creates to implement specific features for your site. This can include themes (the look and feel of your pages) and customized features to meet your business requirements.

> **Extension code**

The Magento marketplace offers extensions to extend the core Magento codebase. In Magento 1 this was an area of significant performance impact as most of the extensions available were of poor code quality. Magento 2 has implemented much more stringent requirements to improve extension quality in the marketplace.

Application Code Recommendations:

> **Use PHP7**

This is a major PHP upgrade which is yielding significant improvements in execution time for the Magento application code. Earlier versions are supported by Magento 2 but PHP7 will deliver the best performance and scalability results.

> **Be Circumspect with Custom Code**

While custom code is necessary, be sure to manage the custom code outside of the Magento core files to leave the core clean. Execution time anomalies will be easier to detect and resolve and upgrade paths should remain unaffected.

Database Performance: Databases are used in several underlying areas of Magento 2:

> **Core Magento DB (Normally MySQL/Relational DB)**

This is the EAV database that stores all product, sales, pricing, order and invoicing details. Magento 2 has improved performance in this area as you are allowed 3 separate master databases in Enterprise edition: Checkout, orders, and product data can each use a separate master database.

> **Session (Normally Redis or other key/value in memory store)**

This is the component that stores detail about user sessions.

> **Object Cache (Normally Redis or other key/value in memory store)**

This component stores frequently compiled blocks of PHP code to enable faster response times when generating HTML documents.

Database Recommendations:

Store session in a memory key/value store. While this can be deployed on MySQL or even in a flat file, high performance is critical to an overall healthy Magento implementation.

Servers Architecture: Ensuring you have appropriately sized instances to execute application code and run your databases is a critical performance consideration. See [Chapter 3](#) for more information and recommendations.

Application Code Monitoring

We recommend that all of the above is monitored by an Application Performance Monitoring (APM) solution. A correctly installed APM tool (such as New Relic) can provide you with a real-time ability to uncover exactly where time and CPU cost is being spent. This is the case even down to a line of code.

With APM, you will be able to see;

- The time being incurred by code execution, whether it is core, custom or extension code which is taking time and consuming resource.
- How many DB calls are being made, the time it takes to execute those calls and their effect on the code execution time.
- The utilization of your server infrastructure so you can understand if the servers are at capacity or not.

Performance Configuration in Magento: Here is Magento's quick guide on performance configuration for Magento 2 (without environment tuning points).

1. Install Magento using web installer: Just pass all steps with default settings using web installation process

2. Turn on performance features (some asynchronous data processing and client side optimizations)

Go to admin. Stores->Configuration->Advanced->Developer:

Grid Settings: Asynchronous indexing: Enable

CSS Settings: Minify CSS Files: Yes

JS Settings: Minify JS Files: Yes

Template Settings: Minify HTML: Yes

JavaScript Settings: Enable JavaScript Bundling: Yes (we can't recommend trying minification of JavaScript now as it is currently being re-worked).

Go to admin. Stores->Configuration->Sales->Sales Emails:

General settings: Asynchronous sending: Enable

//Merge of CSS and JavaScript head files -- should be executed after fixes are implemented

3. Activate flat indexes for catalog

Go to admin. Store->Configuration->Catalog->Catalog->Storefront:

Use Flat Catalog Category: Yes

Use Flat Catalog Product: Yes

4. Switch indexers to asynchronous update mode

Go to admin. System->Index Management:

All indexers should be in "Update on schedule" mode

5. Switch to production mode

php bin/magento setup:static-content:deploy

php bin/magento setup:di:compile

set MODE production manually in web server config

6. Re-index catalog

php bin/magento indexer:reindex //run indexation itself

7. Clean caches

Go to admin. System->Cache Management:

All caches: Refresh

Front-end Performance

Front-end performance is a broad topic with several important areas of focus:

CSS / Javascript: Ensuring quality and size of on-page CSS and Javascript.

- > Magento 2 will provide minification of CSS and Javascript out of the box. Alternate methods of minification can be selected if desired.
- > Earlier versions of Magento may require minification technology deployed on your server or CDN.

Image size / quality: Ensuring that images are not too big and are optimized to fit into the area on the page that they are filling, rather than uploading a large file which is then reduced in size to fit the template

- > Images are compressed by Magento 2 out of the box.
- > Additional image optimization may be required for specific browser types such as mobile or small screen devices.

Network latency: How quickly content is downloaded over the network. This is a key consideration for international sites if your users are in one country and your content is in another. Each request has to pay a penalty of latency that quickly adds up to poor page performance.

➤ International sites in particular should give consideration to use of CDNs for optimization of the TCP and TLS conversations between the browser and your website. See [Chapter 6](#) on how to choose a CDN for your Magento site.

3rd party Javascript: Adding tracking and popup tags onto your website has a significant impact on page load time. It is critical to remove any unneeded tags and optimize any tags that are on your website to ensure they don't execute at the same time as visible page elements.

Serving Traffic Over HTTPS And HTTP/2

Modern browsers will accept traffic over the newer HTTP/2 protocol. Right now, In the USA, HTTP/2 ready web browsers account for 91.17% of all Internet users. HTTP/2 allows the browser to receive more content all at the same time from the web server than was previously possible in HTTP/1.1. HTTP/2 is a major step forward in improving the performance of websites. Our experience with Magento websites and [external studies](#) have shown that serving websites over HTTP/2 is significantly faster than HTTP1.1 and SPDY.

In order to serve your site over HTTP/2 it must be fully encrypted with a TLS (SSL) certificate and all the content should be on the one domain (see earlier notes regarding serving static content for performance). To maximize the performance of your website, one of the easiest things you can do is upgrade to full HTTPS and ensure your delivery infrastructure (host or CDN) will serve your website using the newer HTTP/2 protocol.

In order to get the maximum benefits from HTTP/2 you should not “shard” your static assets onto secondary domains such as “images.example.com”. This old HTTP/1.1 performance technique actually degrades performance. Of course, you still want those static assets driven by a CDN, so select a CDN that offers both full page caching with Varnish Cache and static asset caching. This also means that public CDNs that offer JavaScript library hosting (like a jQuery CDN) will not help the performance of your site.

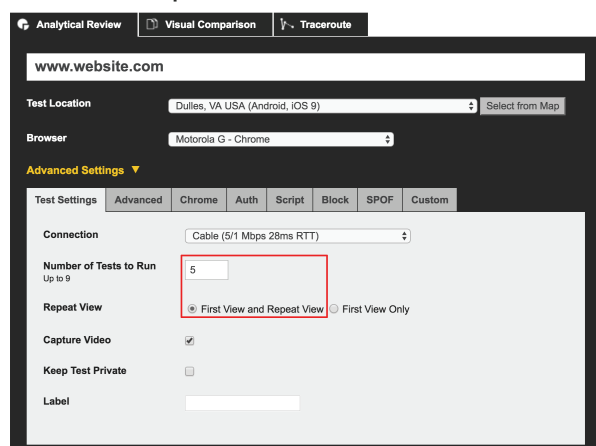
Improving Front-end Performance

Synthetic monitoring tools such as the free Webpagetest (www.webpagetest.org) or the Dev tools in your browser are a good place to start to analyze the front-end performance of your website and see opportunities to improve it. As noted on page 32, Magento has a number of recommendations for both server and client side improvements.

After identifying a particular page from your Magento website which you wish to work on, you can then run some tests in Webpagetest to understand where there might be specific areas in the page available for improvement.

Run about 5 tests (first and repeat view) so you can understand how your caching strategy is working for that page. The multiple first views will help you understand if your delivery infrastructure caching is working and the repeat views help you understand if your in-browser caching strategy is working.

Test a website's performance



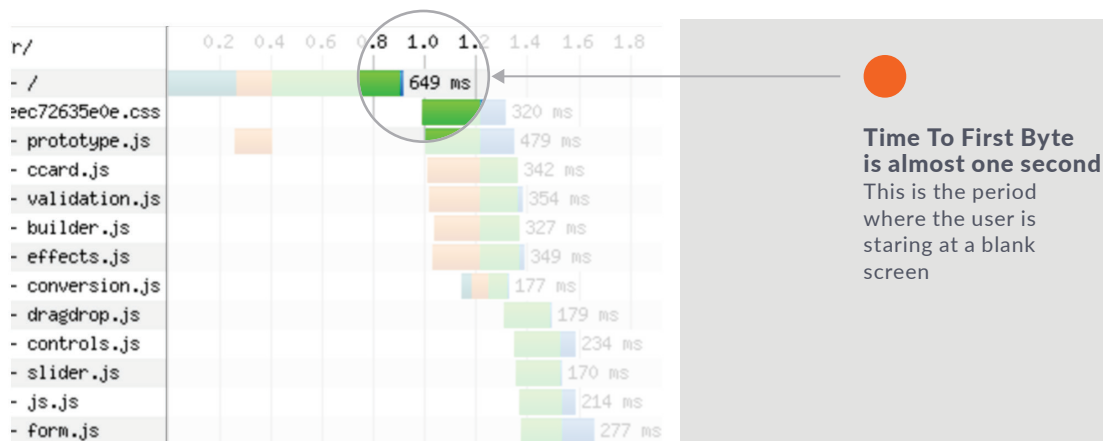
Webpagetest results will then provide some scoring around front-end optimization elements. In the above test results caching, time to first byte, compression and use of Content Delivery Network would improve the page performance.

By clicking on these results, Webpagetest will also provide specifics around these results so you know which assets being served require you to update cache settings or compression:

First Byte Time (back-end processing): 61/100

896 ms First Byte Time
514 ms Target First Byte Time

In this instance Webpagetest shows a first byte time of nearly 1 second which is the period the user would be waiting with a blank screen.



To address this particular finding, you should review the code execution time in your Magento website and/or caching of the HTML document, SSL termination (may benefit from termination closer to the user through use of a distributed delivery system such as CDN) and the DNS lookup time (review your DNS provider).

In addition to measuring adherence to best practice web page performance principles (such as compression and caching), the Webpagetest “Waterfall” can also provide indicators as to whether you have any assets “blocking” the page render.

As noted above, 3rd party scripts inserted into your page (via your custom code) are notorious for slowing down page load. If they are loading earlier in the page waterfall, your user experience of the front-end load time will be impacted. Look for bars on the waterfall which appear to be blocking the next ones coming in (in the same way the HTML blocks loading in the above image).

Caching Considerations

To optimize Magento performance the best available approach is to use Full Page Caching to generate and serve content once and then reuse this content for multiple users.

This approach applies to both static objects (such as images) and dynamic content (such as HTML).

The highest performing Magento sites take a particular focus on ensuring that HTML documents are cached in addition to static content as this delivers lightning fast responses and significant server offload.

How To Cache Static Objects: Magento 2 provides an image caching methodology in production mode so that static view files are not produced on the fly but are created in an asynchronous fashion and stored in a directory on the Magento 2 server. Ideally, these assets will not be fetched from your Magento server directory every time a user requests them. Instead, they should be requested once and served from caching infrastructure which should operate in front of your website.

Caching static objects is a common use case for a CDN. Leveraging a CDN allows the content to be stored as close as possible to your users which delivers a significant performance benefit over caching within your hosting environment.

Implementing a CDN for your website will require a DNS change for your domain so that the traffic requesting your website can pass through the CDN. Head to your DNS provider to make a DNS entry as per the specific instructions of your chosen CDN provider. Some CDN providers will provide DNS hosting as well so you can move your DNS to the CDN provider instead. Don't forget you'll need to set up security certificates so choose a CDN that is HTTPS friendly.

Setting up the CDN to serve static objects should be as simple as making the DNS change and adjusting the CDN configuration to cache and serve the static objects pursuant to their cache control header;

Request #13		Details	Request	Response	Object
13. magento2-		HTTP/1.1 200 OK			
14. magento2-		Date: Wed, 21 Sep 2016 18:17:27 GMT			
15. magento2-		Server: Apache			
16. magento2-		Last-Modified: Tue, 13 Sep 2016 13:44:26 GMT			
17. magento2-		ETag: "1c14-53c63cf9f0c85"			
18. magento2-		Accept-Ranges: bytes			
19. magento2-		Content-Length: 7188			
20. magento2-		Cache-Control: max-age=31536000, public			
21. magento2-		Expires: Thu, 21 Sep 2017 18:17:27 GMT			
22. magento2-		X-Frame-Options: SAMEORIGIN			
23. magento2-		Keep-Alive: timeout=15, max=98			
24. magento2-		Connection: Keep-Alive			
25. magento2-		Content-Type: image/jpeg			
26. magento2-					
27. magento2-					
28. magento2-					

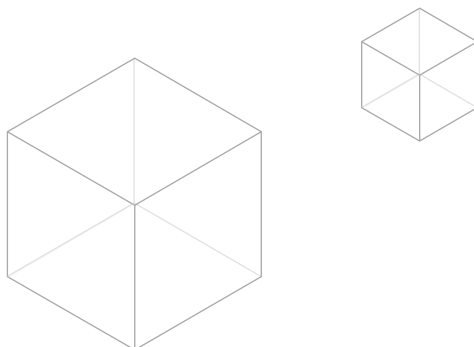
Caching Static Objects for Performance

Historically, legacy CDNs have requested a change of “domain” to push static objects onto the CDN. This would involve serving your static objects from `cdn.YourMagentoWebsite.com` or say `static.YourMagentoWebsite.com` rather than from `www.YourMagentoWebsite.com`. This process was advantageous for performance with old browsers. However, with modern browsers this process is now a performance antipattern.

The modern HTTP/2 protocol allows for “streaming” of assets on one TCP connection and provides for much higher utilization of that connection. Moving assets onto separate domains prevents modern browsers from taking advantage of HTTP/2's high reuse on one connection as the browser must make more TCP connections than necessary.

Keeping your static objects on the main website domain and serving over HTTP/2 can increase your website performance by 10-20%.

You should not move static objects onto a separate domain to implement static object caching. Make sure you **do not set a separate base URL** for Static View or media files in the following configuration screen.



The screenshot shows the Magento 2 Configuration page for 'Base URLs'. The left sidebar contains a navigation menu with categories like 'Web', 'Catalog', 'Customers', 'Sales', 'Services', and 'Advanced'. The main content area is divided into two sections: 'Base URLs' and 'Base URLs (Secure)'. Each section contains several input fields for different types of URLs, with checkboxes for 'Use system value'. The 'Base URLs' section includes fields for 'Base URL', 'Base Link URL', 'Base URL for Static View Files', and 'Base URL for User Media Files'. The 'Base URLs (Secure)' section includes fields for 'Secure Base URL', 'Secure Base Link URL', 'Secure Base URL for Static View Files', 'Secure Base URL for User Media Files', 'Use Secure URLs on Storefront', 'Use Secure URLs in Admin', and 'Offloader header'. A 'Save Config' button is located in the top right corner.

Do not set a separate base URL for Static View or media files on the above screen.

**Note - See below with respect to your Varnish Cache installation options as ideally, you will be able to run your Varnish Cache layer as both the static object cache tier (or CDN) and a full page caching layer.*

Caching HTML

“Full Page Caching” or Caching HTML documents is the best practice for optimal website speed and is implemented by the highest performing websites.

Not only does caching HTML documents deliver your users an incredibly fast site experience, it dramatically reduces the server resources required to run your application. This is a win- win as it both gives your customers a great user experience, while reducing spend needed to run your website.

Out of the box Magento 2 is built to serve HTML documents from cache. This is a significant step forward from Magento 1 and any other ecommerce platform.

As with static caching, HTML documents can be cached locally or in a CDN. CDN caching is the preferred option as it delivers the best overall performance benefit.

Magento 2 includes full page caching capabilities within its native state but recommends leveraging Varnish Cache 4 (an open source caching solution) to store HTML documents. To leverage Varnish for Magento 2 you need to either install it yourself or leverage a hosting partner or a CDN that provides Varnish Cache 4 in its native format. Varnish Cache 5 is being released currently and will also work with Magento 2 as Varnish Cache 5 supports VCL4.

Varnish Cache Installation Options

Install on the Magento Server

You can install Varnish Cache directly on the Magento server. Magento provides detailed instructions on the [installation process here](#). Here are some things to consider with this approach to consider:

For scalability and availability purposes, running your Varnish Cache Instance on your Magento server is not ideal. This installation option provides only one Varnish Cache server so you have no redundancy and your Varnish Cache Instance will be competing for resources with the executing PHP code.

You should be running your site over HTTPS for both security and performance reasons. Varnish Cache will not handle HTTPS connections from your clients and therefore you will need to install a layer (such as a load balancer or other reverse proxy) in front of your Varnish Cache installation to handle the HTTPS connection.

You should explore options to extract diagnostics from the Varnish instances in a repeatable fashion so that you can analyze the cache hit ratios and understand which HTML and static objects are being cached (and which are not).

Install on Separate Servers

Follow the [Varnish Cache installation instructions](#) to set up a Varnish Cache tier in front of your Magento 2 website. What to consider with this option:

This installation option should include more than one Varnish Cache server running in a High Availability configuration. You will need load balancing in front of these instances.

You should be running your site over HTTPS for both security and performance reasons. Varnish Cache will not handle HTTPS connections from your clients and therefore you will need to install a layer (such as a load balancer or other reverse proxy) in front of your Varnish Cache installation to handle the HTTPS connection.

You should explore options to extract diagnostics from the Varnish instances in a repeatable fashion so that you can analyze the cache hit ratios and understand which HTML and static objects are being cached (and which are not). You will need to consolidate the logs from the various Varnish Cache instances.

Run A Cloud Based Varnish Cache Service

Pre-packaged cloud based services (like [section.io](#)) can provide you with Varnish Cache that is ready-to-go for a Magento website including everything you need for Full Page Caching, Static File CDN, scalability, availability, global distribution, TLS (SSL) traffic management, reporting, ease of configuration, metrics and support. For this option you will need to consider:

The ability of the solution you pick to integrate seamlessly with Magento 2.

The cost implications for your website compared to the time saved by utilizing an easy-install version of Varnish Cache.

Where your customers are located. Read more about globally distributed servers in [Chapter 6](#).

Measuring Results

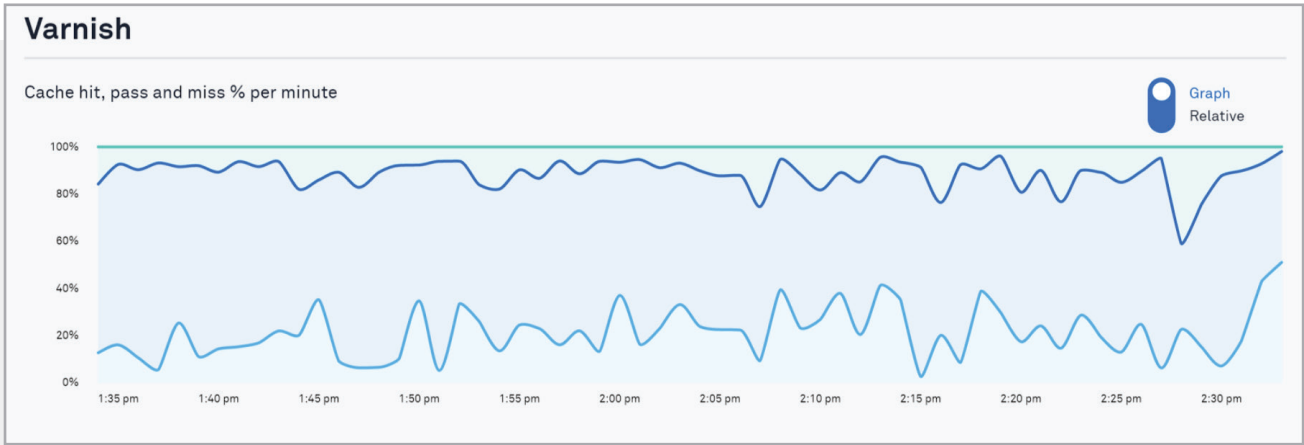
The key to the success of any caching solution is visibility. Without visibility of cache performance (what is being served from the cache and what is going to your hosting environment), it is extremely difficult to tune the caching solution to provide an optimal caching outcome.

You should be able to break up cache performance broadly into:

Static content cache hit rate; and
HTML content cache hit rate

For each of these asset types you should be able to analyze the Hit, Miss or Pass rates by request type. While it is important to cache as much as possible for performance and scalability, it is equally as important not to cache and serve the wrong content. An HTML page which includes user information (such as address, login information or cart details) should not be cached to avoid “leaking the session” or sharing that personal information with other users.

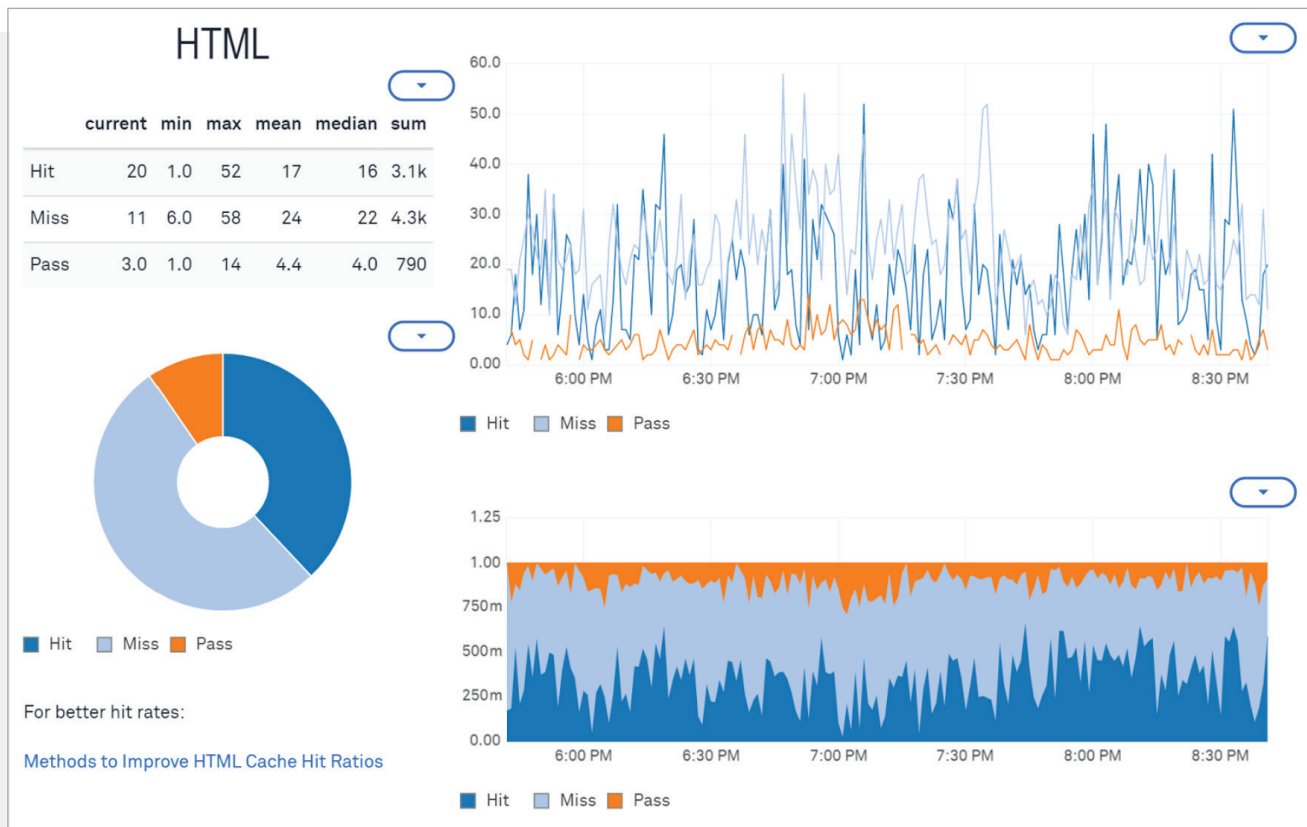
Grouped metrics by asset type can provide indications as to the success of the caching strategy and by viewing trends over time, engineering and operations teams can watch for application deployment moments which may change the “normal” caching hit trend.



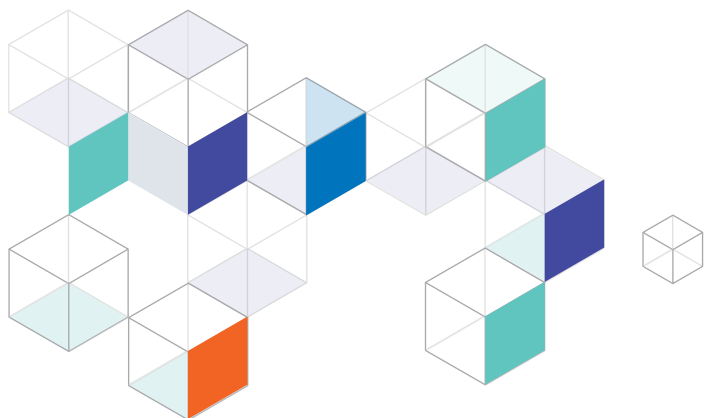
You should have access to logs by request to show for each and every request how Varnish Cache handled the request. If you have access to these logs in real time, you can tweak the configuration of Varnish Cache and test the results quickly to adjust your caching strategy. This fast feedback loop for caching configuration can considerably increase the propensity of a development team to achieve higher cache hit rates (and avoid caching the wrong content).

? uri_path	⊕ ⊖ □ ⚠ /wish'
? useragent	⊕ ⊖ □ ⚠ Mozilla
? varnish_handling	⊕ ⊖ □ ⚠ hit
? varnish_hitmiss	⊕ ⊖ □ ⚠ hit
? verb	⊕ ⊖ □ ⚠ GET

HTML content cache results are a key metric to surface. A single HTML document request can generate 100-200 static content requests. Unless you can split out the HTML component easily, you may not gain visibility and control over your most critical and sensitive item to cache.



If you install Varnish Cache you should consider pushing the Varnish Cache server logs and Varnish traces to log management tooling such as Loggly, an ELK stack or Splunk.



8 | Security Bonus

Protecting Your Magento Website From Attacks



Summary:

Magento websites are a prime target for malicious attacks that could compromise customer data or take your site offline.

Knowing the common types of attacks and how they affect your Magento website is crucial to preventing attacks.

By installing security at various layers throughout your website setup you will be better protected.

The management of the security of your Magento website is important not only to maintain and protect the integrity of your customers' data, but also to ensure the ongoing performance and availability of the website.

Website security is important in order to maintain:

Data Integrity
Business Continuity
Intellectual Property Protection.

Many modern website attacks intend to bring a website offline by overwhelming the compute resources available to that website, or causing challenges and locks in the compute environment or application code execution.

As a very popular and monetized (through ecommerce) website content management system, Magento is a prime target for malicious Internet attacks. Magento website owners should be aware of and able to combat potential attacks ranging from those that siphon off customer payment details, to attacks that hold websites ransom against the threat of taking a website offline.

Common Types Of Attack

DDoS: A Distributed Denial of Service (DDoS) attack is essentially a flood of requests hitting your website from many different locations. The attackers' intent is to deny your real customers access to your website by taking up all the resource your website has to serve those customers. DDoS is differentiated from a DoS (Denial of Service) attack by its distributed nature. A DDoS attack will be launched from many locations rather than one or a few locations.

DDoS is a more difficult form of attack to deal with than DoS attack due to its distributed nature, which makes it harder to diagnose and block the attack. Indeed, many ecommerce websites who experience a sudden rush of genuine shoppers to their website mistake the symptoms of too many customers as a DDoS attack (or vice versa). Both genuine traffic increases and DDoS attacks are essentially a problem of too many requests hitting your website at once, and both result in insufficient resource being available to process many (if any) legitimate customer requests.

OWASP Top Ten: The [Open Web Application Security Project \(OWASP\)](#) publishes and maintains a list of the most common attacks against websites. These types of attacks can and have been instigated against Magento websites. While a number of these styles of attacks are not necessarily directly related to maintaining or improving the performance of a Magento website, they are important to be aware so that you can avoid downtime which may occur when websites need to diagnose and mitigate an attack that is already underway. The OWASP definitions of these attacks are below:

1 Injection

Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

3 Cross-Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

4 Insecure Direct Object References

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

5 Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

6 Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

7 Missing Function Level Access Control

Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

8 Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

9 Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

10 Unvalidated Redirects and Forwards

Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.





Security In Depth

To maintain a well performing and highly available Magento website, it is important to provide several layers of security. We recommend you use the following to ensure your website is protected:

1 DNS Protection

Use of a quality customer-facing DNS solution is important to prevent attackers from overwhelming your website through a flood of DNS requests. Your DNS provider should be able to confirm an ability to handle DDoS attacks.

An even better protection at the DNS layer is to use a provider who can deliver redundant DNS systems for your website so that in the instance of one DNS provider being overwhelmed by any sort of attack, the second provider will be available to continue service for your website. You can set up redundant DNS services yourself or choose a website delivery provider who includes this service for you.

2 HTTPS

Delivering properly encrypted traffic from your servers all the way through to your customers' browsers and back again is a core line of defence in making sure your Magento site and your customers' details are secure.

Deployment and maintenance of a high quality SSL certificate (the certificate you need to demonstrate your site is secure which gives you a HTTPS web address) is important to prevent potential flaws in the encryption which may open the traffic up to interception, interpretation and exploitation. Qualys SSL Labs use a handy rating system to help users discern the quality of their SSL certificate. Lower ratings indicate that your encryption levels may not be satisfactory with respect to areas such as cipher support, protocol support, or key exchange support, or could indicate your certificate is installed incorrectly or not trusted for the domain of your Magento store.

Visit www.ssllabs.com/ssltest/ to test your website certificate rating.

If you wish to immediately enhance your certificate rating, you should address any shortcomings found from this review. You could also investigate the use of certificates issued by your website delivery platform, as they may provide and manage higher rated certificates on an ongoing basis than you are able to secure directly.

An Extended Validation certificate, commonly used by banks or other websites that manage highly sensitive data, is not necessary for your Magento site and will not enhance the security of the web traffic to and from your website. Extended Validation certificates may improve user perception, but do not improve security as they use the same encryption protocols.

3 Network Protection

Attacks can also occur at the networking layer. Your website needs to be able to detect and reject networking style attacks including those at the TCP layer. You should partner with hosting and site delivery providers who provide network-level protection at large scale so that your site is not subject to performance degradation or failure as a result of network attacks.

4 Caching

A well structured caching layer can prevent your core infrastructure and compute resource from becoming overwhelmed by requests for certain assets.

As we reviewed in [Chapter 4](#), caching means an asset or assets can be served from a cache, preferably from an elastic infrastructure which is not part of your core hosting infrastructure. In this way, you can defeat some DDoS attacks simply by having more resource readily available to serve the attacker's requests than the attacker can muster to generate the requests.

Every Magento website should maintain a quality caching tier in front of their web servers for the purposes of both improving performance and scalability of the website directly and for providing an additional security layer. Distributed, elastic cloud solutions will provide the best results for these purposes.

This caching tier needs to be well tuned. For example it is very common to be able to bypass all caching efforts by simply adding random parameters to querystrings. Ideally, all the URLs and their possible valid querystring parameters will be addressed to make sure simple efforts to negate the cache are not trivial.

5 Rate Limiting And IP Blocking

Detecting and blocking requests based on IP ranges or GeoIP Databases can be helpful for certain styles of attack. While some attacks will avoid IP blocking by moving the attacking vector IPs around or attacking from a large and varied range of IPs (such as with a DDoS attack), other attacks can be handled well by limiting the ranges of IP addresses which can make requests on your Magento website. For example, your customer base may be solely from one country and in this case you may wish to block the IP ranges for other suspect countries to avoid the chance that attacks could be launched from those countries.

An additional alternative to all-out blocking of IP addresses or address ranges is to limit the frequency with which an IP address can connect and make requests from your website. Normal customer behaviour usually presents as a much lower frequency of request than a number of different types of attack. This is known as request rate limiting.

By installing the right delivery infrastructure for your website, with very limited effort you should be able to manage the IPs which can connect to your web infrastructure and set upper thresholds for the rate at which any particular IP address can make requests to your web application.

6 Web Application Firewall

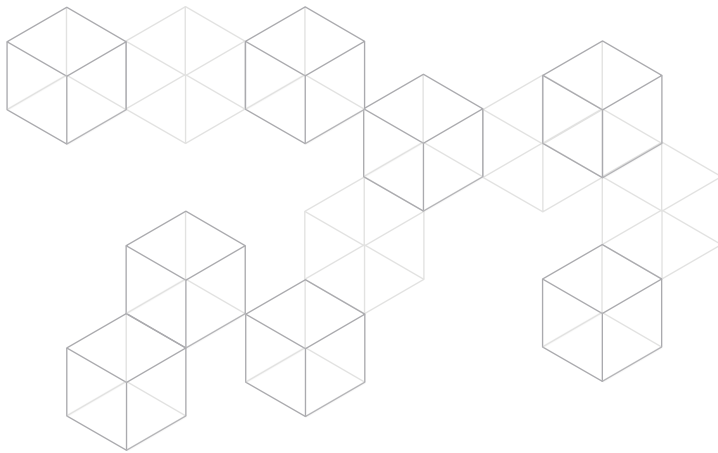
Placing a Web Application Firewall (WAF) in front of your application can be a very effective way of controlling attacks which may occur above the networking layers at the HTTP protocol layer. A WAF can inspect the HTTP requests being sent to your website and, based on a set of rules, determine if the requests may be malicious and block them, or valid and then allow them to continue.

The types of attack a WAF can detect and block include those outlined in the OWASP top ten above. By inspecting, detecting and blocking malicious requests, you can avoid system outages. When websites are compromised by these types of attacks, most often, a website will be taken offline voluntarily by the website owner to avoid the potential calamitous complications of leaked or hijacked customer details and payments information. Installing and maintaining a WAF for your Magento website can prevent these outages.

Beware of one-size-fits-all WAF vendors. These systems are often optimized to reduce the chance of the system breaking the protected application so the vendor can minimize support requests. You'll get a better result with a WAF that is tailored to your application.

Installing and maintaining a WAF for your website can be a complicated matter. You need to make sure you have the right compute infrastructure and the right tooling to be able to view the activity within the WAF and manage the rule sets for your application. A WAF returning too many false positive blocks will cause real customer frustrations and hence become frustratingly useless very quickly. Conversely, a WAF returning too many false negatives (or requests which should have been blocked but were not) will not provide the level of protection which it promises.

Therefore, when installing a WAF, make sure you have the tooling to quickly and simply test the WAF settings in your development and staging environments before turning it on immediately in production. You should have good access to real time reports, metrics and logs for your WAF in addition to flexibility to manage the rulesets and run the WAF in each of your development and staging environments.





Now that you've diligently read every word (right?!) of our guide on optimizing your Magento site for optimal speed, scalability, and security, you're ready to get started on actually making the changes to your website. Don't be daunted by this; by implementing just some of the suggestions in this guide you can quickly see an improvement in various areas of your site.

If you take just one thing from this guide, it is that you should make sure you're utilizing Varnish Cache for your Magento site. Magento has made it quite easy to implement Varnish Cache, and it's a best-in-class caching solution which solves performance, scalability, and to some extent security all at once. If you set up your Varnish configuration so you are caching HTML, managing cookies so more users can be served cached content, and isolating dynamic content, you will be well on your way to a faster, more scalable website.

When thinking about what other optimizations you should make, we recommend taking these steps to determine what changes you want to suggest to your team, and then using our Action Plan to lay out the work, assign resources, and set a timeline.

1 | Measure your current web performance using the tools in [Chapter 2](#). You may already be performing better than you think! This will also help you determine exactly what areas you can improve.

2 | Meet with your team to determine what resources and how much time you're willing to dedicate to a better website. Think about how much ROI you would get out of increased page views and what type of revenue increase you can expect. How much you will improve and increase your revenue will also depend on how you're already performing.

3 | Come up with a timeline for getting the work done: if you need to break it up into smaller sections and tackle one thing at a time, that's fine - it's better to get a few improvements in than delay the whole project because it seems too big to finish at once.

4 | Write your action plan and get started: We've created a template for you below - write specific tasks and meet with those responsible for each overall section of work to guide them as they get started. A lot of this work will likely fall on your development team, so make sure they understand the importance of the end result.

Need help? [section.io](#) is a website performance, scalability, and security solution that makes it easy to configure Varnish Cache for Magento and add additional security features on a globally distributed cloud network. We're the only solution that gives developers full control of their configurations, meaning they can truly optimize their setup to work best with your website and customers. [Contact us for more information](#) or [sign up for a free, 14-day trial](#) with no card required.

Action Plan Template

Title:

By:

OBJECTIVE	MEASURED BY	TASKS	TEAM MEMBER	DEADLINE

Example Action Plan

section.io's plan for improving performance and security
By the section.io marketing team

OBJECTIVE	MEASURED BY	TASKS	TEAM MEMBER	DEADLINE
Improved performance of website	Increase in web page speed of 1.5x or more Revenue increase of 2% or more	Consider CDNs and evaluate ROI	CTO/CMO	October 7
		Implement Varnish Cache with HTML caching	Developer B	October 15
		Check that hosting solution is set up for optimal performance	Developer A	October 15
		Test performance following improvements	Marketer or Developer	October 30
Increased security of website	A or A+ SSL certificate rating	Review DNS protection	Developer A	October 7
		Implement Varnish Cache with or without CDN layer	Developer B	October 15
		Consider if WAF option is needed	CTO	October 15
		Add HTTPS to all pages	Developer A	October 20

About section.io

section.io is the only website performance, scalability and security solution which gives Magento developers complete control over configuration, testing, and global deployment of Varnish Cache and other web enhancement tools.

Unlike legacy CDNs, who lock speed and security tools in fixed networks, section.io provides a user friendly content delivery solution so developers can create a customized web performance and security composition for their Magento site.

Using section.io, leading Magento sites build better websites that deliver faster pages and increased ecommerce revenue.

Features of section.io include:

Global Content Delivery Network

section.io's CDN comprises of globally distributed servers (PoPs) strategically placed along the Internet Backbone so content gets to your global customers faster.

Easy Setup Varnish Cache

section.io allows Magento websites to configure a sophisticated Varnish Cache layer in just 5 minutes. We use an unmodified open source version of Varnish Cache, and you can choose the version of Varnish Cache which works for your website.

Free SSL Certificates and Hosted DNS

section.io manages procurement, installation, and renewal of your SSL certificate for HTTPS for no additional costs. If you manage multiple domains under your section.io application, we can provide and manage SSL certificates for each individual domain. section.io's hosted DNS sets up your application on a network of 24 PoPs across 6 continents.

Local Development Environment

section.io is the only CDN to provide developers with local testing and staging environments so changes can be tested before they are pushed live. Safely configure your chosen proxies in your local development environment to provide the maximum performance and security improvements.

Real Time Logs and Metrics

Get increased visibility into how your caching and security features are performing with real-time logs and metrics.

Contact Us:

Want help optimizing your Magento site for performance and scalability?

Sign up for a section.io account to quickly configure Varnish Cache on a global server network, or if you have more questions feel free to **contact us**.

