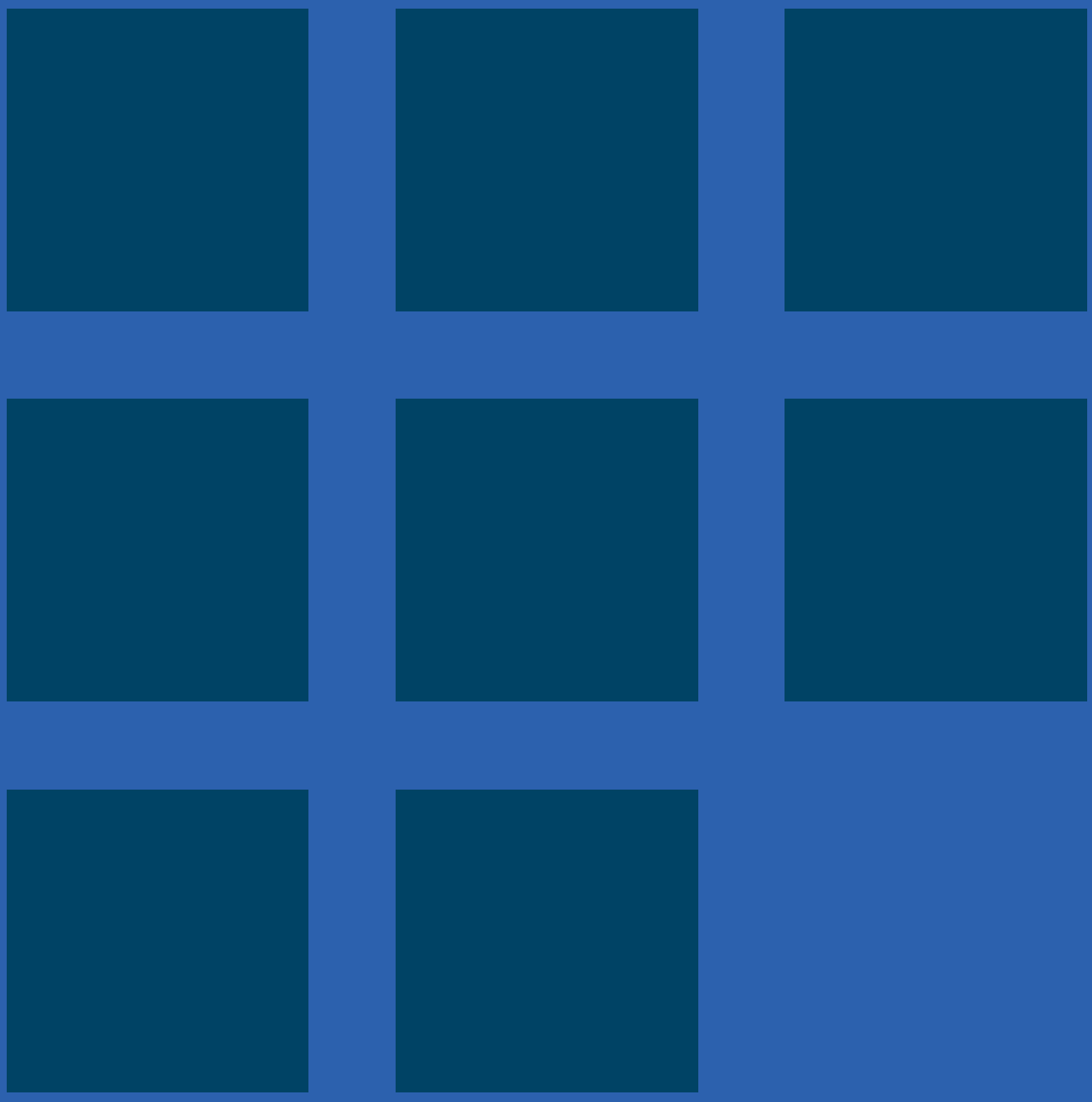

THE
BLUEPRINT
A WEB PRO'S GUIDE TO WORDPRESS

<link>
<code>
1010
0100
0101
01001010010100

bluehost



Bluehost was founded in 2003 with one goal: to make a better hosting company. Built on open source technology, it's since grown to become one of the world's largest cloud-based online solutions providers focused on small and medium-sized businesses. Its experts confidently guide users towards a successful web presence, and its platform now powers millions of websites across the globe, serving the needs of small businesses, bloggers, WordPress devotees, entrepreneurs, and more.

Content Contributed by Bluehost Core Team
Published October 2019

CONTENTS

INTRODUCTION

Building a digital presence goes far beyond building a website with text, images, and a contact form — and WordPress is a platform designed to go well beyond a simple website.

WordPress is a perfect foundation for simple sites and blogs, as well as eCommerce stores, social networks, email newsletters, online courses, and custom platforms. You can start small and have the potential to expand into amazing digital possibilities, and you connect WordPress with other tools and services that make up a digital presence.

As your WordPress sites grow, the simple workflows and hosting you started with may need to adapt and evolve with you.

Enter The Blueprint: A Web Pro's Guide to WordPress, our playbook for making fast and secure WordPress sites.

In this guide, we'll explore the essential skills and workflows web professionals like you need to succeed online. You'll discover how technologies work — and how to make them work for your online goals. And you'll get insights on how to migrate and troubleshoot WordPress sites like a pro.

SECTION ONE

PERFORMANCE

01

WHY PERFORMANCE IS IMPORTANT

Optimizing the performance of your WordPress sites is critical to their success. Just as the design and aesthetics of a site are important to the overall impression a visitor has of a business, performance is equally vital to this perception.

Numerous studies have shown that conversions, page views, and overall customer satisfaction go down as the time it takes for a page to load goes up. Every second added to the load time on your site risks losing visitors.

In addition to being vital to the customer experience, performance matters for getting traffic to your sites as well. In fact, in 2018, Google began using page speed as a ranking factor for mobile searches. If SEO is important to you, then the speed of your sites should be, too.

There are three primary areas that can affect the performance of a web page: the front end of the web browser, the server, and the network. There are things you can do in each area to help improve the speed of your sites.

Front-End Performance

The web browser is where a request for a web page originates and ends. The code that executes inside the browser and tells the page how to display is typically referred to as the "front end." And it's one of the first places where you can improve performance.

Request Size

There are several causes for front-end sluggishness, but the most important is size. Simply put, the more data that has to be transferred to the visitor, the longer it's going to take. Limiting the amount of CSS, JavaScript, images, and videos that are loaded on a page is one of the most effective ways to reduce load time and improve performance.

The fastest site possible wouldn't load anything but HTML, but of course that wouldn't be particularly attractive or functional. So you need to reduce the size of those additional assets that are loaded with each request.

CSS and JavaScript

CSS and JavaScript should be minified to eliminate whitespace and comments in the code. You want to remove anything that is not necessary for functionality in CSS and JavaScript. Typically, this is handled during the development process using command-line build tools.

Here are a few popular options you can use:

- **UglifyJS2:** This is a popular JavaScript parser, minifier, compressor, and beautifier toolkit.
- **Closure Compiler:** Built by Google, this tool will parse and rewrite your JavaScript to make it more efficient.
- **cssnano:** This CSS compression tool can also rewrite and remove unused code.

PRO TIP

There are also websites that allow you to copy and paste CSS or JavaScript and they will minify the code for you.

Well-designed themes and plugins from responsible developers will already have minified their assets in the built version that you install on your sites. In the event that they haven't, some caching plugins and CDNS will handle this minification for you, which we'll cover later in this section.

IMAGES

Images are the most common cause of large page sizes. Huge, high-resolution images may look fantastic, but if they are not compressed properly, they can be massive in file size. They will take longer to load for a visitor, especially on slower connections.

When adding images to your sites, it's important to remember two main rules:

1. Compress the original image first to reduce overall size.
2. Reduce the dimensions of the image to match what will actually display on the site. You don't want to load an image that is many times larger than the space it will actually display in.

You can compress images using image editing software before uploading them to your sites or utilize one of many online tools to do it for you. Several of the most popular even have WordPress plugins that will do the compression for you when you upload the image to your sites. Two popular options are WP Smush from WPMUDEV and Compress JPEG & PNG images from the team at TinyPNG. TinyPNG also allows you to upload an image directly on its website.

If you're building a theme that uses background images or similar design methods, you should ensure you use the proper dimensions for the element. For images inserted into the content, WordPress helps with this through its automatic handling of responsive images. It will add `srcset` attributes to allow browsers to request the image with the most appropriate dimensions based on the size of the viewport.

VIDEOS

Videos tend to be pretty large because they are essentially a series of images with audio. Entire books can be written about video compression methods, but the easiest way to improve video performance on your site is simply to take advantage of external video hosting. Popular third-party video hosting services such as YouTube and Vimeo will handle compression and fast delivery to viewers without blocking or slowing down the loading of the rest of your content.

Page Design

Lazy Loading Assets

When you need to load a lot of images on a single page, it's better not to request them all at one time. The visitor's browser window is limited in size, so loading images that are far down the page and outside of the viewport can lead to slower overall load times. The process of delaying the loading of images until they are about to be visible on the screen is called lazy loading.

Some themes and plugins may include this tactic within their designs. There are also plugins that will add this functionality for content within the body of posts and pages, such as Jetpack's Lazy Load module.

Render-Blocking JavaScript

The traditional method for adding JavaScript to HTML documents is through the use of a script tag in the head section, which is called render blocking. It looks like this:

```
<head>
    <script type="text/javascript" src="example.js"></script>
</head>
```

The problem with render blocking is that it requires the browser to request, load, and execute this JavaScript file before it can continue rendering the rest of the document. To improve the overall perceived performance of a page load, it's important to make scripts that are not critical to the initial rendering of the page asynchronous or deferred through the `async` and `defer` attributes.

The `async` attribute tells the browser that it is ok to continue loading the rest of the content while the script downloads simultaneously.

The `defer` attribute will cause the browser to wait to execute the JavaScript file until the rest of the document has been downloaded.

DID YOU KNOW

Though defer is widely supported by modern browsers, it does not work reliably in Internet Explorer prior to version 10.

Server-Side Performance

The second place where you can make the most difference in performance is in the code that executes on the server where your sites are hosted. While front-end optimization helps reduce the time it takes for a page to render once it is retrieved, optimization of the Hypertext Preprocessor (PHP) programming language can help build the response and deliver content to the visitor's browser more quickly. PHP executes the database queries that are run on an individual request. This is typically referred to as "time to first byte" or TTFB. We'll discuss how to measure this in the tools section later.

PHP

The first thing you can do to optimize PHP performance is to ensure you are running the latest version. PHP performance has been steadily improving with each major release. At the time of this writing, version 7.3 is the most recent release.

Once you're running the latest version, the next thing to consider is the more PHP code that has to execute, the longer it will take. This does not necessarily mean that having more active plugins is bad for performance. One hundred tightly scoped, well-written plugins may perform better than a single monolithic, unoptimized plugin. However, the more plugins that are added to a site, especially larger, feature-rich plugins, the greater the chance that you'll run into PHP performance bottlenecks.

If you are building a plugin or theme, it's important to optimize it by limiting execution to only the portions of code needed for that request. In WordPress, this means choosing the right action hooks and adding conditionals to prevent unnecessary code from running.

Reading reviews and choosing from trusted developers can help ensure that you've selected well-optimized plugins and themes for your site.

Database

If a PHP script seems slow, it is often due to the database queries it is executing. Things to look out for are running a large number of queries, queries that gather large amounts of data, or queries that collect data from multiple tables.

When building a site, even if you aren't writing the underlying code, there are a couple of ways to ensure you are not executing queries that might be a performance drag.

Paginate Query Results

When you return a list of objects on a page, such as posts or products, you should always break the results up into pages. It might be fine when you only have 10 items, but when that begins to scale up to hundreds or thousands, you'll start to see major issues. If you're coding the theme or plugin, then you should avoid using -1 as a `posts_per_page` value in `WP_Query`.

Avoid Meta Queries

Custom fields are great for storing additional metadata on posts, but if you query posts based on those values, it searches an unindexed column in the postmeta table and then combines those results with the posts table. These queries are notoriously slow. It's better to use taxonomies for this purpose when possible.

Network Performance

There isn't much that can be done to speed up the delivery of your sites across the network lines that make up the internet. The one thing that can be done is simply to have it served from as close to the visitor as possible.

CDN

Typically, your site's code and database will only originate from one primary data center, but you can utilize a content delivery network (CDN) to help serve assets and cached results from another datacenter that might be closer to your visitor.

There are two free CDN options that are quite effective:

1. **Jetpack Site Accelerator:** This plugin can automatically upload your images, CSS, and JavaScript to Jetpack's global network of servers, cache them, and serve them to visitors.

2. **Cloudflare:** Cloudflare goes a step further. You can route all your traffic through Cloudfare's global servers, which cache everything and serve the traffic to your visitors. Distributed denial of service (DDoS) protection comes with Cloudfare's free plan.

Server Performance

Caching is one of the easiest but most important things you can do to speed up your websites. WordPress is a dynamic content management system. Without any caching, every page is put together on the fly for each request that comes in. Caching allows your server to save some of the results of these requests, so the next time they are needed, it doesn't have to build them again.

Types of Caching

There are four primary types of caching that you can use in a typical WordPress request, each of which can provide benefits to your site.

Browser Caching

When a visitor requests a page, it will download additional assets, such as CSS, JavaScript, and images. Unless instructed not to, browsers will typically save copies of these assets so that the next time they are needed by a page, they won't need to send a request across the network and download them again.

This is done through the usage of `Cache-Control` headers. These are used by the browser and server to determine if assets should be cached and for how long before they expire and are needed to be requested again. Your hosting provider may set these, or they can often be controlled through a caching plugin, which we'll cover in this section. On Bluehost, our built-in caching system will set these headers for you based on the caching level you have selected.

To determine whether they have a cached version to use, browsers identify assets using their URLs. Sometimes, you may need to update some CSS or JavaScript on your site to ensure that users see the latest version the next time they visit. To help with this, WordPress allows the use of a version parameter on `wp_register_script()` and `wp_enqueue_script()`. This appends the supplied version number to the end of the URL for an asset, so when it changes, the browser will see that it is different and request a new copy.

Page Caching

Full-page caching is the caching method that has the biggest performance benefit — but it comes with some drawbacks. This method takes the fully built response from the server and saves it in RAM on the filesystem or in a CDN, such as Cloudflare. The next time a request for that page is sent, it does not need to execute PHP or database queries and the result can be returned directly back to the visitor.

Full-page caching can't be used in all situations. If your site relies on dynamically serving unique content to each visitor, you likely won't be able to take advantage of full-page caching. For instance, if your site is a membership site where visitors need to be logged in to view content or participate, then their requests can't be cached and served to another user.

Many hosting providers offer some form of page caching. At Bluehost, our built-in caching solution can handle page caching for your site if you select Normal, Advanced, or Aggressive in the caching control section.

Alternatively, you can achieve full-page caching with a caching plugin or a CDN, such as Cloudflare or Sucuri.

Object Caching

Earlier in this section, we noted that database queries can be a major cause for TTFB slowness. Object caching is a way to reduce the need for those expensive queries. The results of a query that takes a long time to execute can be saved, then the cached value can be retrieved instead requiring the query to run again.

More advanced object caching methods include using an external, persistent object cache, such as Redis or Memcached. These store query results or other data in RAM and allow for very fast retrieval.

WordPress has a method of built-in object caching through its Transients API. The default method is to store the expensive query results in the `wp_options` table for faster lookup the next time they are needed. Some plugins can hook in and override this to store these objects externally, typically in RAM.

Bytecode Caching

Every time PHP executes, it has to be compiled into machine-readable operation code, or opcode. For scripts that have not changed, it is faster to cache a query result so it does not need to be compiled again the next time it runs. This is done through the use of the PHP OPcache extension.

PRO TIP

OPcache will most likely be installed and managed by your hosting provider, but it's good to be aware of its existence.

Caching Plugins

Whatever type of caching you choose to take advantage of, it's important to use a supporting caching plugin to ensure timely purging of your caches. These plugins can help identify when to cache things and when to invalidate a cache because the content or design has changed.

If you're a Bluehost customer, we have caching built in by default, as well as a 'must use' plugin (mu-plugin) for automatically handling the cache invalidation.

There are several other good caching plugin options as well:

WP SUPER CACHE

This plugin, originally developed by Automattic, handles full-page caching by storing the response as a static HTML file. It's very fast and simple to set up and, as the most popular caching plugin, it is in use on over two million sites.

W3 TOTAL CACHE

This is an extensive caching plugin that can cover all levels of caching. It does full-page caching through static HTML files, and can integrate with an external object cache for offloading database queries. It can even coordinate directly with CDNs for faster delivery of your site's assets. The settings can be overwhelming, and it's light on documentation, but it covers a lot and is used by over one million sites.

CLOUDFLARE

This plugin is only needed if you are routing your site traffic through Cloudflare. This will enable the cached pages to be purged from your WordPress site when you update content. It's listed here because of Cloudflare's reliability and as an alternative to page caching handled at the server level.

Diagnosing Performance Problems

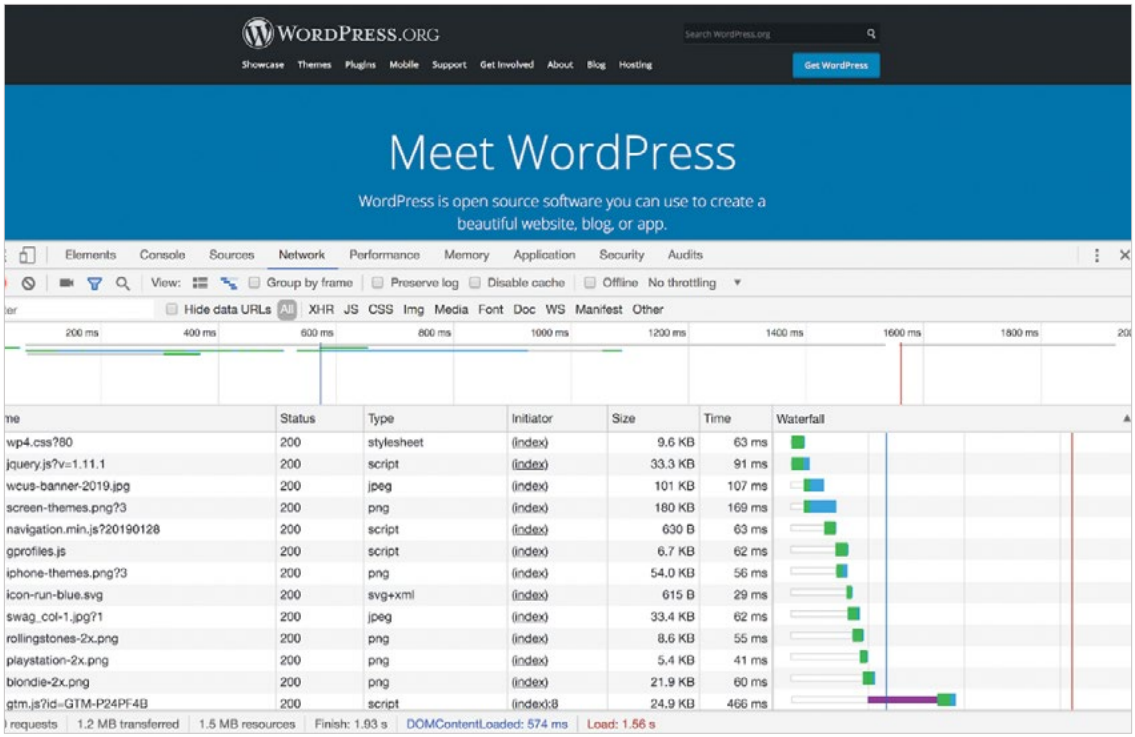
Now that you understand the places where you can optimize your sites to improve performance, it's important to figure out how to analyze the sites' speed and determine where the bottlenecks might be. Here are a few tools that are helpful in troubleshooting and diagnosing performance-related problems.

Browser Developer Tools

Most modern browsers include built-in tools for doing all sorts of troubleshooting and analysis. Depending on the browser, there are varying tools included, but a common one is the Network tool. This will show you every HTTP request sent in the process of a page loading and provide additional information, such as the size of the response, how long it took, and more.

Here are a few helpful things you can do using the Network tool in the browser:

- Determine if there are specific requests that are taking a long time to load.
- Verify the size of individual responses and assets being loaded.
- Analyze whether your assets are being cached by the browser or not.
- See TTFB or waiting time to understand if the performance of a site is impaired by slow server-side performance.



Here's how to access the Network tool:

- In Google Chrome, you can open the developer tools by opening the main menu and selecting **More Tools** > **Developer Tools**
- In Firefox, select **Toggle Tools** from the **Web Developer** menu (under **Tools** on OS X, Linux, and Firefox on Windows).
- In Safari, enable the developer tools in the **Advanced** section of **Preference** by checking **Show Develop** in the menu bar. After that, you'll see a **Develop** menu where you can choose **Show Web Inspector**
- In Microsoft Edge, press the F12 key, or right-click and select **Inspect Element**

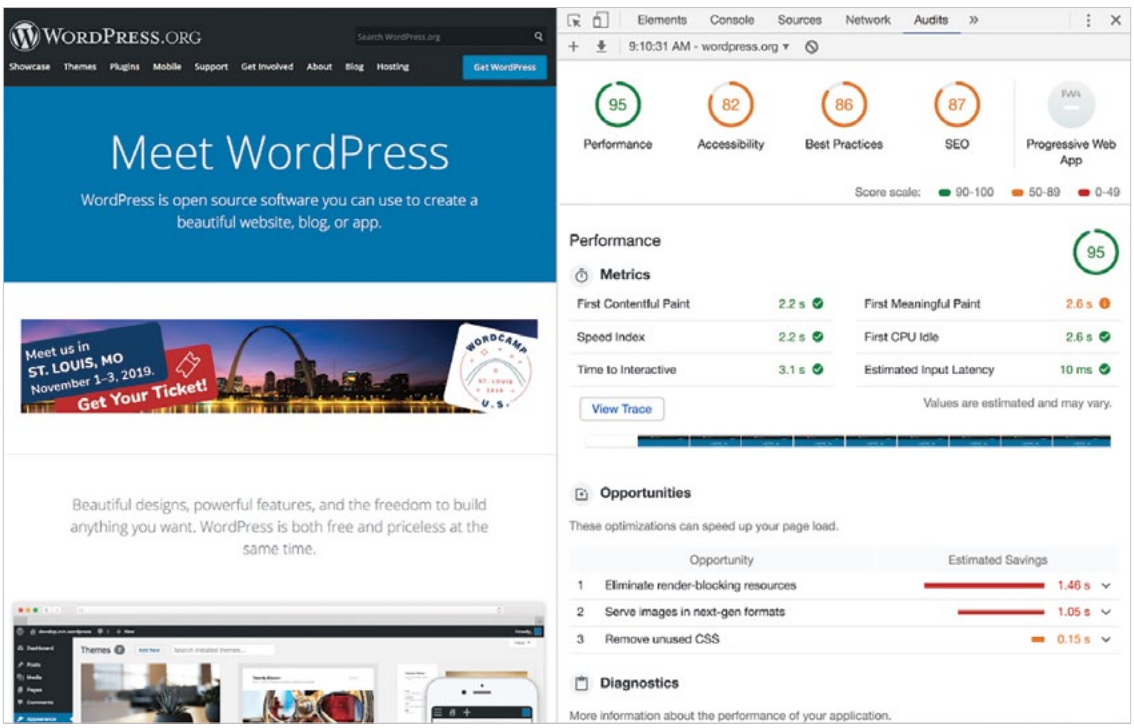
Google Lighthouse

Inside Google Chrome's developer tools is an **Audit** tab that can run an automated analysis of a website using Google's Lighthouse tool. You can choose to run the test from the perspective of a mobile or a desktop request, as well as a simulated slow internet connection.

The analysis will generate scores in performance, accessibility, best practices, and SEO. This is helpful to get a general idea of how well your site is built. Don't focus on the number in the score as much as the overall range it falls in.

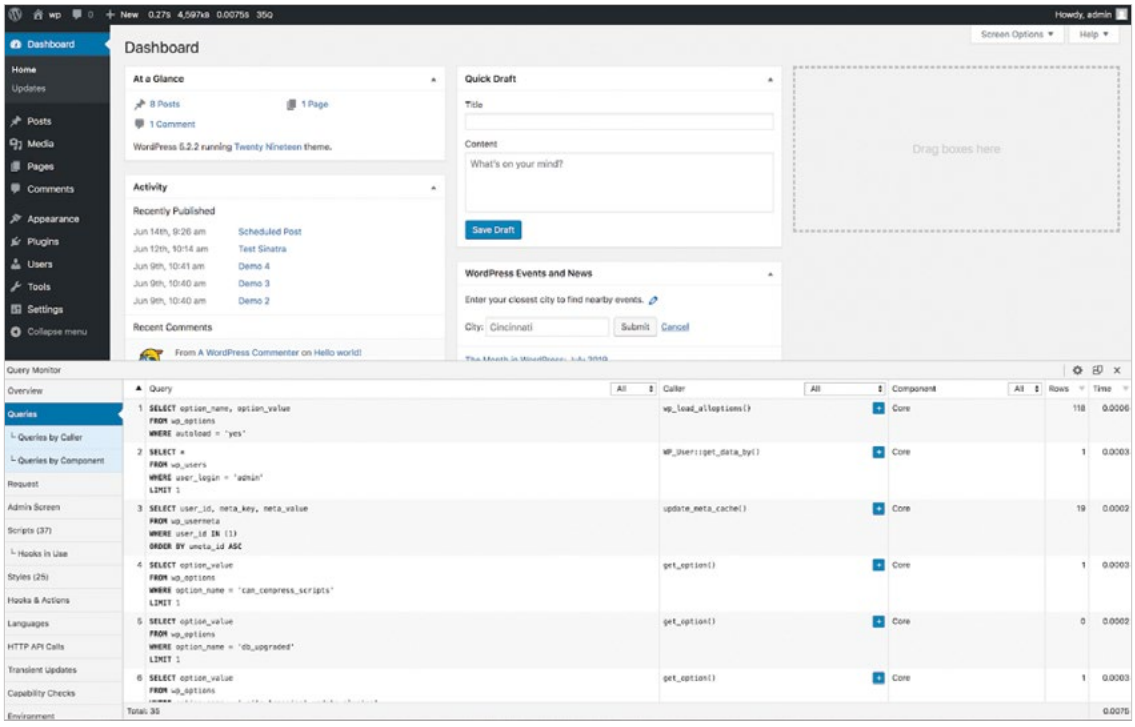
In addition to the scores, Lighthouse also provides details about performance timing metrics, opportunities where you can improve performance, diagnostics of where some other slowness might be occurring, accessibility issues, and SEO improvements that can be made.

A similar browser agnostic tool is GTmetrix; however, it just focuses on performance and doesn't offer the accessibility or SEO analysis.



Query Monitor

If you've determined there is a TTFB issue from the use of your browser developer tools, you can dig deeper by using the WordPress plugin Query Monitor. It is described as "the developer tools panel for WordPress," and it even looks a lot like the built-in browser developer tool.



Query Monitor provides extremely detailed insight into all the database queries that execute on an individual request, such as where they originated from (theme, plugin, or core), how long they took, and how much data was returned.

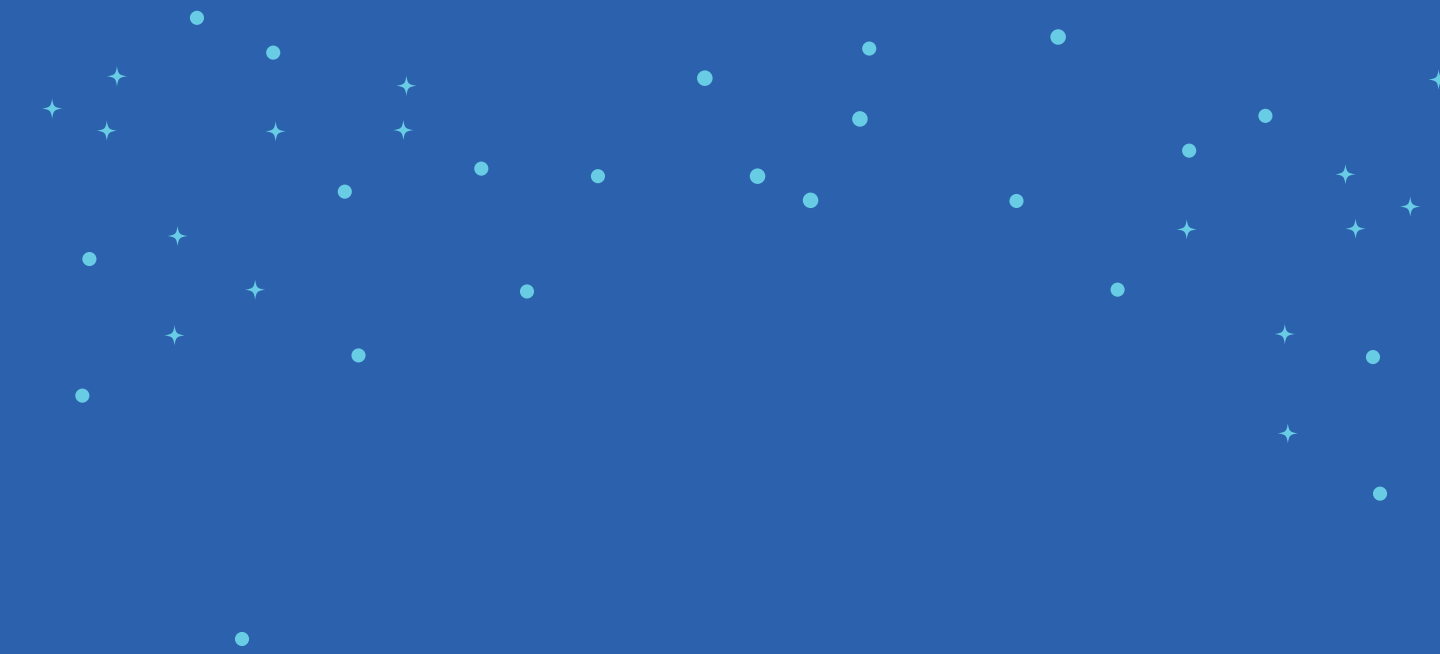
DID YOU KNOW

You can also see which WordPress hooks were used and how much PHP code executed on the request was used by which plugin.

SECTION TAKEAWAYS

In summary, here is a checklist you can use to improve the performance of your sites.

- ○ Minify CSS and JavaScript to minimize whitespace and comments in the code.
- ○ Compress or reduce image size to maximize page load speed.
- ○ Ensure you are running the latest version of PHP.
- ○ Avoid running a large number of queries, queries that gather a large amount of data, or queries that collect data from multiple tables.
- ○ Try to avoid meta queries.
- ○ Utilize a CDN to serve assets and cached results from another datacenter closer to your visitor.
- ○ Find a supporting caching plugin to ensure timely purging of your caches.
- ○ Choose a browser developer tool to help diagnose performance-related issues.



SECTION TWO

SECURITY

02

SECURING YOUR WEBSITES

Security is one of the most important aspects of a website.

Because new vulnerabilities are always being discovered and best practices are always changing, security can be quite intimidating and requires regular attention and upkeep. Making sure your sites are secure ensures the safety of your users, your clients, and your business.

In this section, we'll cover some easy steps that you can take on every site to improve the overall security and avoid the stressful situation of a hacked site.

Themes and Plugins

Choosing a theme and deciding which plugins to install on your site can easily be overwhelming. At the time of writing, the WordPress.org directory has over 55,000 plugins and thousands of themes. This does not include the uncountable number of free and paid plugins and themes from third-party developers.

So, how do you choose which themes and plugins are safe to install from a security standpoint? Although it is not an exact science, here are some things to consider that can help you choose safe and secure options.

Investigate the Author or Publisher

Who created the plugin or theme? This can tell you a lot about how secure a plugin or theme is.

Here are some things to look for:

- Are the authors or publishers active, well known, or reputable within the community?
- Do they have multiple plugins or themes?
- Do they answer posts in the support forums?
- Do they contribute to the WordPress project in some way?
- How large is their user base of active installs?

Each of these factors will have different levels of importance for different people and projects, but they should give you a sense of who an author or publisher is.

Are They Maintained?

In general, it is a good sign if the plugin or theme was updated recently and updated regularly. This shows that the author or publisher actively maintains it and would likely address any bugs or issues that come up. Sometimes there is no need for the plugin to be updated regularly, such as a plugin that performs a very specific task. In this scenario, the feature may not need iteration if the underlying API doesn't change.

Free Versus Freemium Versus Premium

All plugins and themes in the WordPress.org directory are free to install and use. Others can be mostly free (as in "freemium") but some advanced features require you to purchase a subscription or license. In most situations, this is not a bad thing. Paid upgrades usually provide many valuable features, often include premium support, and ensure the author or publisher receives some money to continue developing and maintaining the product.

Downloading from Trusted Sources

You always want to make sure you're downloading code from the WordPress.org repository or an official distribution source for a premium product. These are locations the WordPress project or the developer are monitoring. Third-party repositories or stolen copies of premium products can contain malicious code or they may not update the software with security patches, putting your site at risk.

PRO TIP

Unmaintained software can become a security vulnerability over time as new exploits are discovered. When vendors show signs that they maintain their software, it demonstrates that they care about their users and will work to fix any security vulnerabilities discovered.

SSL Certificates and Firewalls

Secure sockets layer (SSL) certificates and firewalls are essential security features.

SSL Certificates

An SSL certificate is a must-have for every website today, especially because there are free ones available. These digital certificates are used to create an encrypted connection between a browser or user's computer and a server or website.

When a connection between a server and browser is established without an SSL certificate in place, any data, such as emails, passwords, and credit card numbers are sent through these connections as plain text. And as such, this information is accessible to anyone who is eavesdropping on traffic on a network. This is a type of man-in-the-middle attack and most commonly happens when using public, unsecured Wi-Fi networks.

You'll know a website has a valid SSL certificate in place when you see a padlock icon in the browser's address bar. Many users have a trained eye to identify and avoid submitting any information to sites that do not have this padlock. By having an SSL certificate in place on your web server, you provide a more secure experience for your clients and their customers.

If you are a Bluehost customer, you're in luck! Every new WordPress site is automatically configured for SSL with a free Let's Encrypt SSL certificate.

Firewalls

A firewall is a security system that acts as a barrier between trusted and untrusted traffic. Having a firewall is common on networks, but there are also firewalls available for websites called web application firewalls (WAFs). WAFs provide protection against common security issues, such as DDoS attacks, zero-day exploits (which disclose security vulnerabilities immediately after discovery), brute-force hacking, and more.

Because companies that manage WAFs are in the security industry, they are hyper aware of new security vulnerabilities and implement patches to provide protection against them very quickly. This helps ensure that your site is secure against all the latest and greatest attacks, even if you are not able to update your website immediately.

There are many plugins and services that offer a WAF feature. Cloudflare and Sitelock are two popular ones.

Passwords, Password Managers, and Multi-Factor Authentication

Strong and Unique Passwords

Basic best practices for security should always be used on your websites. Using strong passwords is a must to prevent sites from being hacked in brute-force attacks. These attacks are where bots try password after password until eventually guessing correctly. The longer your passwords are, the more possible combinations there are, and the longer it takes to guess.

Passwords should never be reused on multiple sites. It is very common for hackers to take exposed usernames and passwords from compromised sites and set up scripts to test those credentials on thousands of popular sites. If you reuse the same password on multiple sites, hackers would easily be able to log in as you on each of them.

Password Managers

Password managers make using long, secure, and unique passwords on every site easy. A password manager stores your login credentials for sites and secures them with a single password that you remember. No more reusing the same password to prevent forgetting it!

Multi-Factor Authentication

To add another level of security to your site, you could enable multi-factor authentication. The most common form of multi-factor authentication is known as two-factor authentication (2FA). Using a combination of two different factors, 2FA confirms people are who they claim to be through something they know, something they have, and something they are. With more factors come more hurdles for attackers to overcome, so for particularly sensitive sites, you may want to add more authentication factors.

In the real world, a good example of this can be seen when using an ATM machine. In order to perform a transaction, you must produce a physical card (something you have) and enter the correct PIN number (something you know). For websites, 2FA is usually implemented by requiring a password and a time-sensitive, six-digit number provided by an independent service, such as the Google Authenticator app.

There are several plugins in the WordPress.org repository that allow you to add support for 2FA to your site, providing an additional level of security to prevent someone from gaining undesired access to your site.

User Roles and Capabilities

When granting someone access to your website, it's important to only give people the level of access that they require, even if they are trusted. For example, if the person only needs to write posts, it does not make sense to give him or her access as an administrator. Even if people are trusted, limiting what they can do on a site protects it as a whole in case their user accounts are compromised in the future.

By default, WordPress has several roles with varying capabilities.

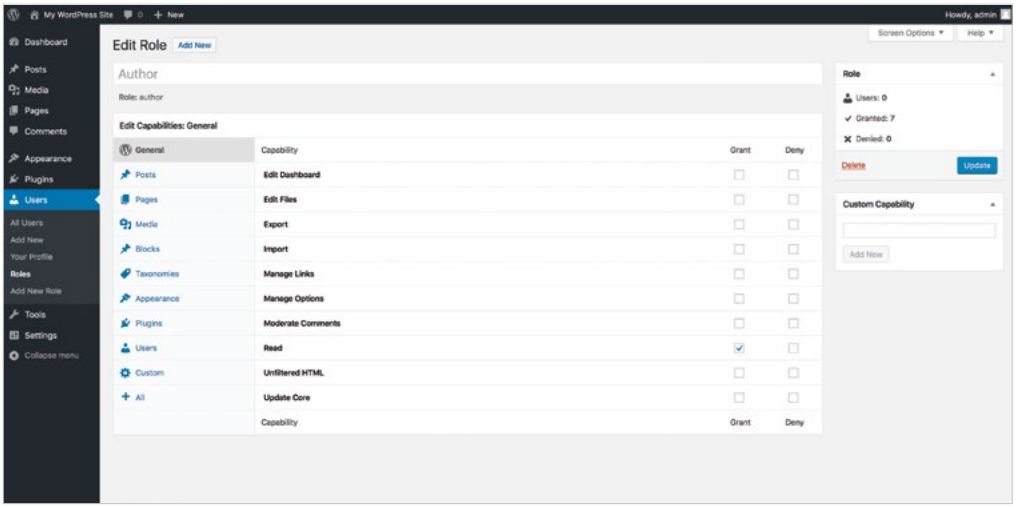
- Subscribers can read content.
- Contributors can create content, edit their own posts, and submit posts for review, but they can't publish content.
- Authors can do everything above, plus upload files, publish posts, and edit their own published posts.
- Editors can do everything above, plus moderate comments, manage taxonomies, edit other people's posts, delete posts, and manage private pages.
- Administrators can do everything.

PRO TIP

There are many plugins that create new, custom roles for specific tasks related to the plugin functionality.

For example, WooCommerce creates a shop manager role for managing products and orders only and a customer role for people who make purchases and manage their orders.

There are also plugins that give you fine-grained control over the capabilities for each role in WordPress.



File Editing

In WordPress, there are two areas of the admin dashboard where files can be edited: the **Plugins** > **Plugin Editor** and **Appearance** > **Theme Editor** screens. These screens can be useful for making quick edits to files. But in some situations, this providing file editing access may be a security risk.

For example, say your company is an agency that manages sites for customers. Any of your administrators who can edit theme files could break the site. Or, maybe you have a continuous integration pipeline set up that deploys changes from a repository. In this scenario, changes would be erased if the files are updated using FTP, or if the changes are deployed automatically, because none of those changes would be tracked.

If an administrator account is compromised, for example, these screens could allow someone to add malicious code to a plugin or theme file.

```
define( 'DISALLOW_FILE_EDIT', true );
```

To add a level of security, these screens can be disabled by defining a constant in your `wp-config.php` file.

This will ensure files can only be changed if someone has FTP access, or if they have access to the project's repository that is configured to deploy to the server. Someone using a compromised administrator account would be unable to inject malicious code into plugin and theme files through the admin.

Attack Checklist

Having a plan or checklist to follow if your site is hacked is important — just like having an evacuation plan for emergencies in your home or office. But just like a fire escape plan, you should create this when your site is up and no one is panicking. Write it up while you are calm and collected.

This will look different for everyone, but the checklist should include things like:

- Where are backups stored?
- How can the site be restored?
- Who should be contacted or involved?
- How can you investigate how the attacker exploited the site?

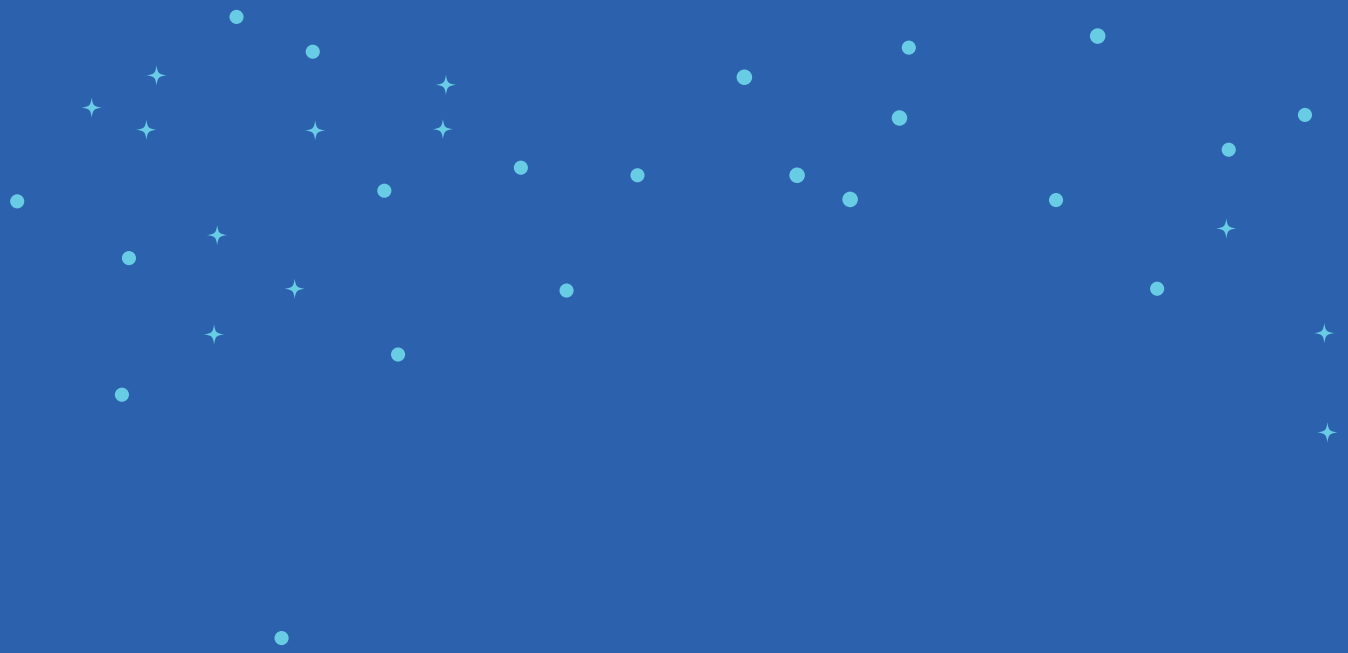
PRO TIP

Having a site hacked can be very stressful, but having a plan in place to deal with it can go a long way.

SECTION TAKEAWAYS

In summary, here is a checklist you can use to improve the security of your sites.

- ○ Use regularly updated and maintained plugins and themes from reputable authors.
- ○ If using a paid theme, make sure the purchase location is reputable.
- ○ Always have an SSL certificate.
- ○ Use a WAF to help improve site security.
- ○ Use strong, unique passwords on every site.
- ○ Enable multi-factor authentication whenever possible.
- ○ Only give users the access level they need.
- ○ Create custom roles to scope user access when necessary.
- ○ Disable file editing in the WordPress admin.
- ○ Create an attack mitigation checklist in case your site is hacked.



SECTION THREE

WORKFLOWS

03

MAXIMIZING WORDPRESS WORKFLOWS

WordPress has used the phrase "Code Is Poetry" for many years. It means that there isn't one perfect way to write code and while there are some rules, code is more of a personal expression than an exact science.

Workflows, or how you go about building and maintaining your WordPress sites, are equally personal and unique, and directly related to the task at hand and the hands taking on the task. As a developer who is paid to build a website or a WordPress product, while you might be delivering code and a website configuration as a final product, what you're really charging for is unique workflows. It's all about the journey, not the destination.

In this section, we'll explore how to use workflows in WordPress, including how to use staging and local development environments, design a backup and recovery strategy, and safely update software.

Staging

Staging environments are complete replicas of production environments, or your live websites. A staging environment contains all your WordPress files, plugins, themes, and uploaded media files, as well as a full copy of the WordPress database containing your posts, users, site options, and other data.

Using a staging environment in your workflow gives you a playground where you can tweak, modify, and safely break things without affecting your live website. Sometimes, the best way to learn how something works is to take it apart and put it back together.

A staging environment is designed to give you confidence in the changes you're making, whether those changes are to the site design, features, or content. Once you get staging exactly how you want it, you can push those changes to your live production website with one button.

Some users choose to write all their content in staging, and some use staging to test a new theme or plugin, while others leverage staging to validate that core, plugin, or theme updates won't break a site.

Create a Staging Environment

Here's how to create a staging environment in Bluehost:

1. Login to the admin dashboard
2. Click on Bluehost logo in the upper-left corner
3. Click the **Staging** tab
4. Click on **Create Staging Site**
5. A message will appear saying *"Your staging environment is ready!"*

How Bluehost Staging Works

When your staging site is ready, the Bluehost **Staging** page will show your staging URL, the staging directory on your server, and the creation date. The staging URL will be a subdirectory in your current site that's hidden from search engines and the public internet.

The **Go to Staging Site** button will automatically sign you into your staging site. Your staging website looks exactly like your production website, so to identify staging, look for one of two giveaways that you're in your staging environment:

1. On the admin dashboard, there will be a red box that says **Staging Environment**

2. In the browser's address bar, look for **/staging/xxxx**, when xxxx is a stand-in for a random number in the URL following your normal site URL.

Below the **Go to Staging Site** button are two options:

1. **Clone Production to Staging**: This option will re-copy all your files and data from your production website to staging. If you've updated your production website since your staging site was created, this will carry those new updates over to staging. It will also overwrite any changes you've made in staging.

2. **Destroy Staging Environment**: This option will completely delete your staging environment, requiring you to click **Create Staging Site** to use staging again.

Using the Staging Workflow

When staging is used correctly, it helps you ensure confidence in the updates you're making and saves you time. Used incorrectly, staging can cause a headache or, worse, it can wipe out important data.

Use this workflow between your production and staging websites:

ALWAYS START FRESH

Before working in staging, it's always a good idea to **Clone Production to Staging** to make sure you're working on the latest files and data. You will first want to check you have nothing left in staging that you'd be wiping out.

LEAVE PRODUCTION ALONE

Once you've cloned your staging website, try not to make any changes to your production website — pushing your staging site to production could wipe out these changes. If need be, start from the top.

MAKE CHANGES IN STAGING

Whether you add or adjust content, switch themes, install updates, or tweak settings, you'll want to make your changes in staging. Confirm that your changes accomplish your goal without any unintended errors or side effects.

SAVE RESTORATION POINTS

This step is optional but highly recommended if you're making multiple changes in staging. In your staging site dashboard, go to the **Staging** page. At the bottom of the page are **Restoration Options**, where you can click on **Add Restoration Points**, allowing you to drop markers you can return to if you break something or want to undo changes.

PUSH STAGING TO PRODUCTION

In your staging site dashboard, go to the **Bluehost Staging** page. In staging, this panel will present three options, instead of the two in production:

1. **Deploy Files Only**
2. **Deploy Files & Database**
3. **Deploy Database Only**

If you're unsure which deploy to use, it's best to **Deploy Files & Database**. This will copy the specified pieces of staging over to your production website, overwriting and deleting anything that wasn't in staging. This is why it's critical to only deploy staging when you're positive production doesn't have new changes that could be wiped out, such as changes from other editors, administrators, or users in the form of comments.

With a large website, it can be faster to only deploy files or the database if you're certain a full deploy isn't necessary.

You can leave your staging site up after a deploy and continue using it to test changes. However, one way to be absolutely certain you aren't pushing old data or files from staging into production — or wiping out new production files or data — is to always destroy staging after making changes. This forces you to re-create staging each time you want to make changes. It may take a few extra minutes but it can also give you extra confidence in your workflow.

Use a Local Development Environment

A local development environment is software running on your computer to run WordPress instead of relying on a server on the internet. It's a great tool for building and testing sites before running them on a live site if you don't want to use a staging environment.

Websites running on your computer are inherently faster than websites running on the internet that need to be downloaded to your computer. The main advantages of local development are faster page loads, page refreshes, and saving of files. These small speed increases add up during the active development of a site and keep you in your development flow instead of constantly changing and waiting for a remote server.

Local development can also make it easier to work on sites without an active internet connection or where internet speeds are slower, like Wi-Fi in a cafe or on an airplane.

There are two styles of local environments: one that relies on your computer's operating system and one that creates a virtual server with its own operating system and software. If you're unsure which is right for you, it's best to use the second option: a local environment that creates a virtual server and doesn't modify your computer's operating system. These virtual server tools are a tiny bit slower than tools using your native operating system but they are still much faster than using a staging environment on a remote server.

When creating a local environment with a virtual server, some tools will prompt you for a PHP or MySQL version to install. Whenever possible, you want your local environment to match the versions being used on your production server. To identify these versions in WordPress, go to the **Tools** menu and select **Site Health**. Click the **Info** tab and open the section labeled **Server for the PHP version** and the **Database** section to identify the database server version.

Here are some local development tools you could use for creating a virtual server:

GUI STAGE TOOLS

- DesktopServer
- MAMP (Mac) / XAMPP (Windows)

COMMAND-LINE STAGING TOOLS

- VVV
- Lando
- Trellis
- Docksal

Backup and Recovery

Backups are critical to protect your data. Whether your clients' website are a source of income or simply a hobby, backups help protect them in the event the sites get hacked or go down. Backups also give you a rollback point in the event you make undesired changes or create an error that you can't resolve.

There are two key things that must be backed up in order to fully restore your site:

1. All files for your plugins, themes, and uploaded media items (such as images, PDFs, and MP3s).
2. A full copy of your WordPress database for each site.

However you choose to back up your site, it's important you know how to restore your site from a backup. We recommend writing a disaster recovery plan well before you need it, so you can follow your methodical plan instead of making haphazard or snap decisions while you're stressed out, frustrated, or emotional.

The WordPress Importer Plugin, found in **Tools** menu on the admin dashboard under **Import and Export**, is *not* a comprehensive backup solution and shouldn't be relied upon for a full site restore.

In some Bluehost plans, comprehensive backups are included, while other plans offer backups as an add-on. These plans are backed by our guarantee, and our support agents are always available to assist you with them.

Many plugins — both free and premium — are also available for comprehensive backups. The most popular is called UpdraftPlus. It will back up and restore your site from popular file storage services including Dropbox and Google Drive. While UpdraftPlus and other backup solutions are used on millions of sites, including on many of our customers' sites, the key difference is they don't carry our guarantee. If there's an issue, we'll always do our best to help, but we can't guarantee the completeness and quality of third-party tools, or be held responsible if they don't work as expected.

When creating a backup and recovery strategy consider these factors:

THE FREQUENCY OF CHANGES AND UPDATES

eCommerce websites change often, so they may need to be backed up daily, multiple times a day, or even hourly.

WHERE BACKUPS ARE STORED

Backups shouldn't be stored on the current server in the rare event it crashes or your site is hacked.

APPLICABLE PRIVACY LAWS

Under European Union (EU) privacy laws, such as General Data Protection Regulation (GDPR), there are restrictions on how websites can record and retain data about EU citizens. These laws apply to all EU citizens, regardless of where the website or organization is located, or where an EU citizen interacts with your site. These laws include data retained in backups, so be mindful of how long you keep backups and whether your backups contain personal data that could fall under GDPR.

Software Updates

Updating software is critical to maintaining a safe, secure, and issue-free website. Updates also can add new site functionality for you, your clients, and their customers.

But updates can also have unintended consequences, as they can potentially break parts of your admin dashboard, the display of your website, or, worse, they could corrupt your data.

Depending on your confidence in an update and comfort with risk, you may either choose to run updates on your production site, or run test updates first on your staging or local environment. You may choose to run all your updates simultaneously or run each individually.

However you update, we strongly recommend making a backup of your files and data before running and update.

Here is a recommended, best-practice strategy for updating software on your sites:

1. Create a backup of your production files and data.
2. In a staging site, open your **Updates** page and briefly review the **changelog** file, noting mentions of new features, security patches, and bug fixes.
3. Run the updates, confirming that admin pages continue to load and visitor-facing features continue to work as you run each update. Try to test functionality mentioned in the **changelog** file and review any design changes or added features.
4. Once you are confident of the updates in a staging site, deploy staging to production, or run the updates from production. While it may take longer to clone staging into production, it also means you can be confident there aren't any differences in versions and that what you validated in staging is exactly what's being pushed to production.

Software updates should be run frequently, regardless of how often you update your website content or whether or not you think you're a target for hacking.

Often hackers aren't specifically targeting a website but crawling the internet looking for vulnerabilities to exploit, like out-of-date and less secure software.

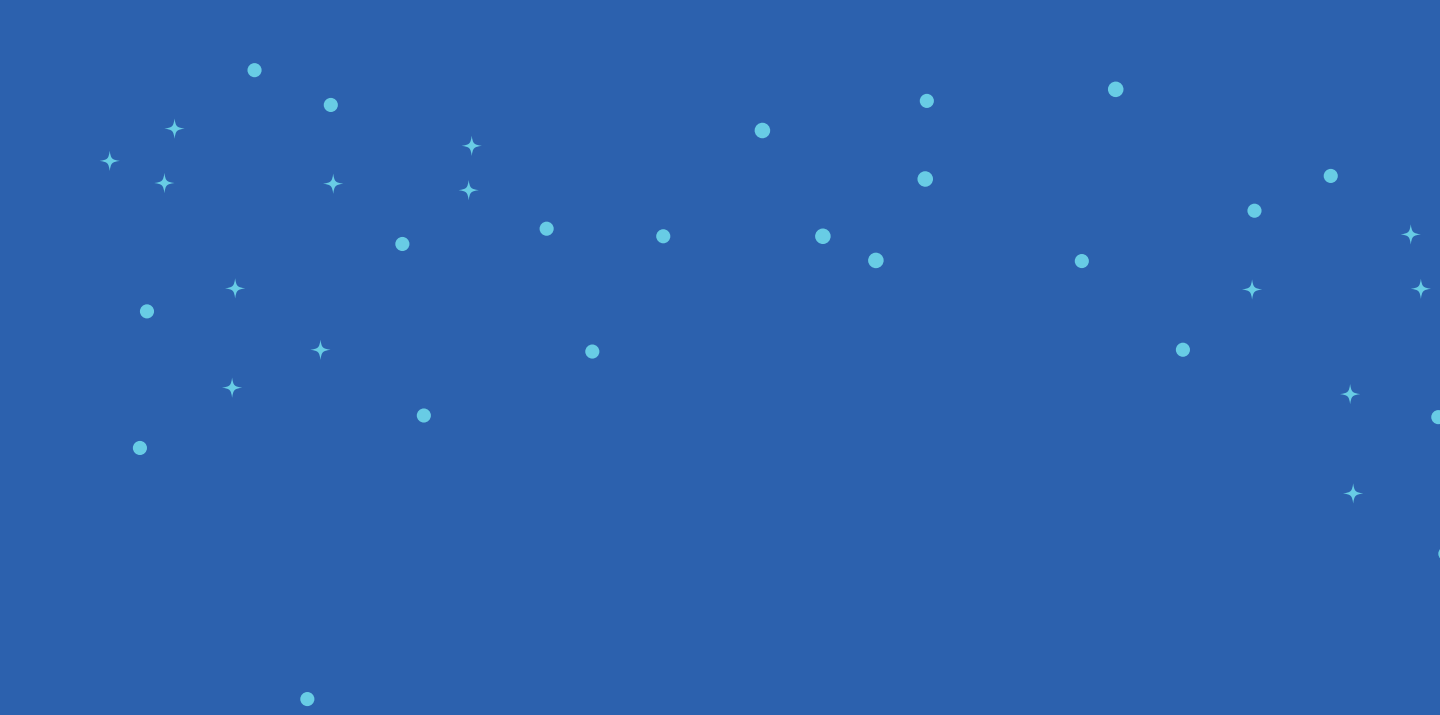
Ideally, updates should be checked every few days or a few times a month, as they are critical to keeping your website secure to protect sensitive data about you, other site users, your clients, and their customers.

We recommend setting an ongoing calendar event in your digital organizer with reminders to run updates.

SECTION TAKEAWAYS

In summary, here is a checklist you can use to improve your workflows.

- ○ Use a staging environment to tweak, modify, and safely break things without affecting your live website.
- ○ Use a local development environment to work on sites without an internet connection.
- ○ Write a disaster recovery plan that includes backup and recovery steps.
- ○ Update your software to maintain a safe, secure, and issue-free website.



SECTION FOUR

TROUBLESHOOTING

04

IDENTIFY AND RESOLVE ISSUES

Troubleshooting is the act of tracking down an issue and resolving it.

Sounds simple, right? However, there are a lot of different ways you can track an issue down, identify the root cause, and find out if it's the symptom of a deeper issue — all of which will be covered in this section.

Troubleshooting a website is much like solving any problem. As a WordPress developer, you understand the inner workings of your sites, and as such, you can leverage that experience to fix issues that come up.

In the following section, we will cover how to properly document your website issues and provide you with a general mental framework you can use to diagnose. We will also discuss common sources of issues and how to take an appropriate course of action.

Over time, your experience with different types of issues will grow and you will almost instinctively know how to handle common problems. Just for good measure, the last part of this section walks through a few specific issues using the resolution processes you are about to learn.

Documenting Issues

The first step in troubleshooting is to identify and document the issues. You'll want to consider what changes have been made on the site recently, who else has access to the site, and when you started seeing a particular issue. From there, you can identify the issue and create a plan for remediation.

When documenting an issue, you'll want to include both essential details, as well as contextual ones.

Essential Details

You will realize your website needs attention when you discover it didn't do something you expected it to do, or conversely, it did something you didn't expect. Either way, the actual results you are seeing don't line up with results you expect or want.

The way you define website issues is always going to be relative to what you consider to be normal. However, some people's version of normal can be a bit unclear. For this reason, it is important to not just document the issue or abnormality that occurred, but also what you consider to be normal and what you expected to happen.

The reality is that some issues only show up under certain circumstances, so you will need to identify those circumstances. For instance, does the issue occur every time you load a particular webpage? Or, do you have to go to that webpage, fill out the form, and hit the **Submit** button? If you aren't clear on the steps to reproduce the problem, you won't be able to communicate the issue to others, nor will you be able to properly validate when the problem is resolved.

As such, it is important to document everything you did to cause the issue. But play around a little bit and see if the problem still occurs if you eliminate, change, or even add a few steps. If a problem has really got you stumped, gaining more clarity and arriving at the bare minimum required to replicate the issue can provide some very helpful clues.

Make sure you include these three essential aspects as you document issues:

1. The steps to reproduce the issue.
2. What you were expecting to happen.
3. What actually happened.

Contextual Details

Keep in mind that so far you've only described the symptoms in the essential details. You still need to take into account some of the context surrounding those symptoms. Sometimes, there is context built into the steps you laid out to reproduce the problem, such as a specific URL you have to visit, a specific section of the site that is impacted like the header, or all the archive pages. However, the more context you can find and document, the better.

Here are a few important things you should take into account when looking into the context of an issue:

WORKAROUNDS

Sometimes knowing how to circumvent the problem can provide enough insight to make the true cause of the issue readily apparent.

TIMING

If you know the exact time that the issue occurred, you can often correlate this with known changes made around that time. Often, this reduces the amount of work necessary to pinpoint the cause.

CHANGES

If you know what changes were made on the site and when, you can correlate this to when the problem first occurred. For example, if the problem occurred right after installing a specific plugin, the problem most likely is with that plugin or with a plugin that conflicts with that one.

SOFTWARE AND VERSION NUMBERS

It is important to know exactly what software you are running and their versions. Obviously, you are running WordPress. Make a note of what version of WordPress you are running. Likewise, make a note of what your active theme is and its version, as well as any plugins you are running and their versions. In some cases, plugin and theme authors will publish a list of software that is known to conflict. If you happen upon one of these lists, it will be handy to have all the details ready so you can do a quick cross-check. Additionally, if you reach out to a plugin or theme author, it is very common for the author to ask for a list of what themes and plugins you have running.

ENVIRONMENT

Certain issues require knowledge about the server or environment in which the website is running. These details include things like what version of PHP or MySQL you are running, what type of web server you are using, who your web host is, and if you have SSL enabled.

While this seems like quite a bit of work to collect all this information, there are a few tools that can help expedite this process, which we've listed below. The only exception here is identifying the workarounds and when the problem occurred; those are manual tasks.

PRO TIP

When communicating with others remotely regarding an issue, use annotated screenshots or record a quick screencast of you replicating the issue. This makes sure everyone is talking about the same thing and prevents delays when addressing an issue in the event the other person misunderstood what you were saying.

Tools for Documenting Issues

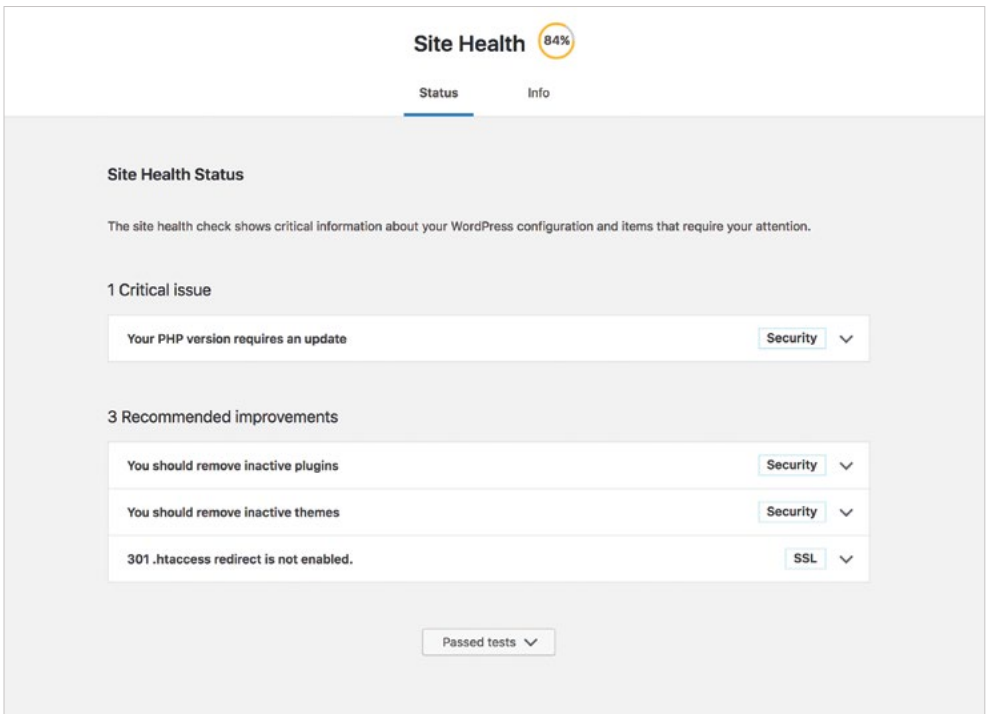
Stream

Stream is a plugin that can help you track the activity on your website. This handy plugin will track things like when WordPress was last updated, who has logged in recently, what content was changed, what plugins were activated or deactivated, what theme settings were changed, and much more. The only thing it can't track are things that happen outside of WordPress, like someone editing or deleting a file on your server, updating your PHP version, moving to a new web host, or things like that. Typically, if those types of actions are occurring, you are more likely to remember them. For everything else, Stream will track it.

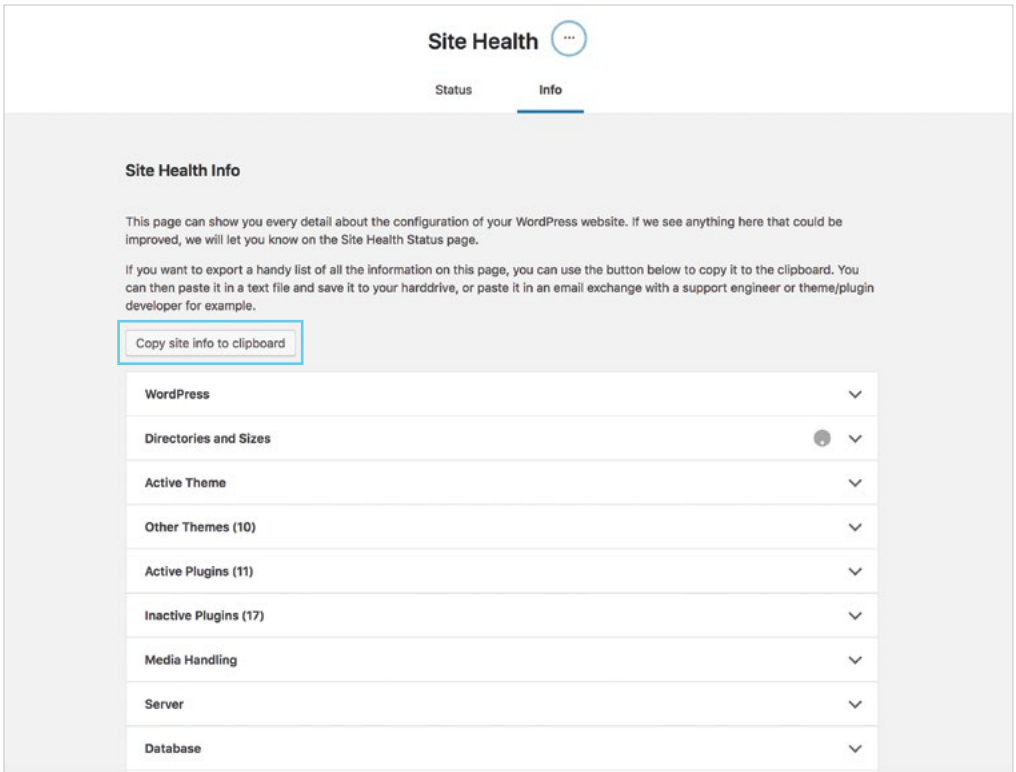
Site Health

As of WordPress 5.2, Site Health is a new feature that keeps tracks of all the details on your environment, listing all your themes and plugins, tracking software versions, and the like. The benefit of this tool is two-fold.

First, the **Site Health Status** page shows a list of issues or recommendations that WordPress, your theme, or active plugins have automatically detected. If you find yourself in a troubleshooting scenario, visiting this page first could potentially save you a lot of time and hassle. Make it a habit to check in on this page from time to time and do your best to address anything that requires attention.



The second benefit is the **Site Health Info** page, which displays a list of environmental and configuration details for your WordPress site. Even better, there is a handy **Copy site info to clipboard** button that makes it easy to copy and paste all your site details in plain text. This can be very helpful when communicating with plugin and theme authors.



Diagnosing Issues

After identifying and documenting issues, it's time to start diagnosing them. The act of diagnosing an issue typically requires using a scientific approach that goes something like this. First, given the data you've already collected, you can form a hypothesis. Next, you can formulate a series of tests to try and either prove or disprove your hypothesis. These tests will require you to do things that could potentially break your site for visitors, result in a loss of configuration, or create other detrimental and potentially irreversible situations. In most cases, however, the side effects of these tests are only temporary and easily reversible.

Precautions

Before starting the diagnosis process, you'll want to be cautious when running these tests. But as long as you've taken the proper precautions — like staging and backing up — there is no need to be afraid.

Staging

The ideal place to troubleshoot is in a staging environment, which was described in Section Three. As you may recall, a staging site is simply an exact copy of your live website that is not publicly accessible. If you break a staging site, all you need to do is sync it back up with your live site and you are ready to start testing again.

Backups

Another option if you don't have a staging site is to create a full backup of all the files and data on your website before you start troubleshooting. See Section Three for backup instructions.

Types of Issues

Before jumping in and trying to determine the source of a problem, get in the habit of classifying the type of issue you are dealing with. Doing so will provide additional clarity and could significantly speed up the time to resolution. While this classification could still be considered part of the documentation phase, we've added it as part of the diagnosis phase since it can occasionally be subjective.

There are many ways that issues could be classified, but we will focus on a classification system that will nudge you in the right direction as you are looking to determine the cause of an issue.

Visual

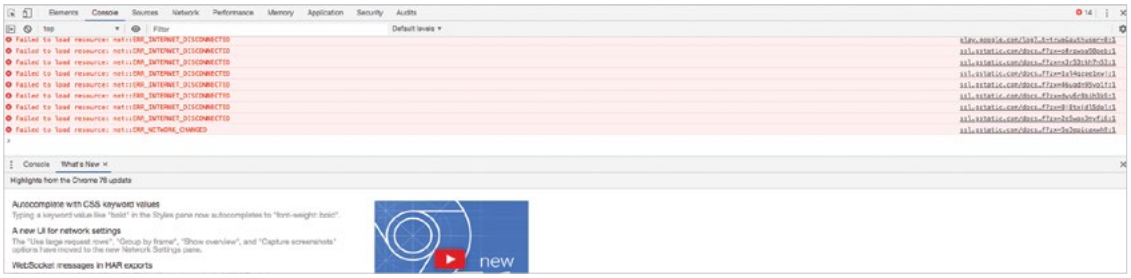
If something generally looks out of place but all the data is correct and functionality works as expected, the problem is often visual in nature. This type of issue typically involves your theme or a page-builder plugin.

Interactive

After loading a webpage, it is common that certain elements on the page will have some level of interactivity. For example, you may be able to click a button to submit a form, hover over an icon to read a description, or drag something from one area of the screen to another. However, if one of these expected interactions no longer works, the issue most likely is with the interactivity of that page. Keep in mind interactivity issues will have an impact anywhere that interactivity is used on your site.

Interactivity issues are most commonly related to the JavaScript code on the page. It isn't uncommon to have a JavaScript issue prevent other non-related interactions from working. Check the JavaScript console in your web browser to see if any errors appear. To find the console in Chrome, click on **View** in the top menu, drop down to the **Developers** menu item, and then select **Console**. Alternatively, you can use the **Command + Option + J** keyboard shortcut on Mac or the **Control + Shift + J** shortcut on Windows. Any errors in the console will appear in red. Keep in mind that you may need to perform the action that would normally initiate the interactivity in order to trigger any errors in the console.

If you do see errors, often these will provide hints as to the true cause of the problem. If the issue doesn't pertain to animation on the page and you don't see any errors, it could be that a specific action is triggering a new request behind the scenes and what you are actually dealing with is a functional issue.



Functional

A functional issue pertains to the ability of a specific webpage, resource, or the site as a whole to load. If a webpage won't load or only loads partially, you are almost certainly dealing with a functionality issue, where you'll see either a blank white screen or a section of the webpage missing entirely. For example, you may see that the footer isn't loading, a calendar or menu doesn't appear, or anything along those lines.

For classification purposes, a functional problem can be triggered by nothing more than visiting a specific webpage. This is almost exclusively related to the PHP code running in a theme or plugin. Again, you can take a look and see if there are any error messages that might provide more clues by debugging your site, or by checking PHP error logs, both of which are described below.

If you go poking around looking for errors in PHP, keep in mind that PHP classifies messages based on their significance. There are notices and warnings, which probably should be fixed but most likely aren't breaking anything on your site. What you want to look for is anything labeled as a fatal error; these are the ones that break sites.

To enable the debug mode in WordPress, you will need access to the WordPress files on your web host. To access them, go to your web host's file manager interface or use SFTP (secure FTP). You want to find the `wp-config.php` file and locate the line of code that contains `WP_DEBUG`. If it exists, make sure the value is set to **true**. If it doesn't exist, just add this line after the `<?php` line:

```
define( 'WP_DEBUG', true );
```

If you do use this method, you should change the word **true** to **false** after you are done debugging. Showing errors on your live website is a security issue as it may provide hackers with enough data to compromise your site.

You can also look at PHP error logs provided by your web host. Web hosts often locate their error logs in very different locations, so you should Google the name of your web host and the term **PHP error log** (or just reach out to your web host directly) to find out where these are located.

Data

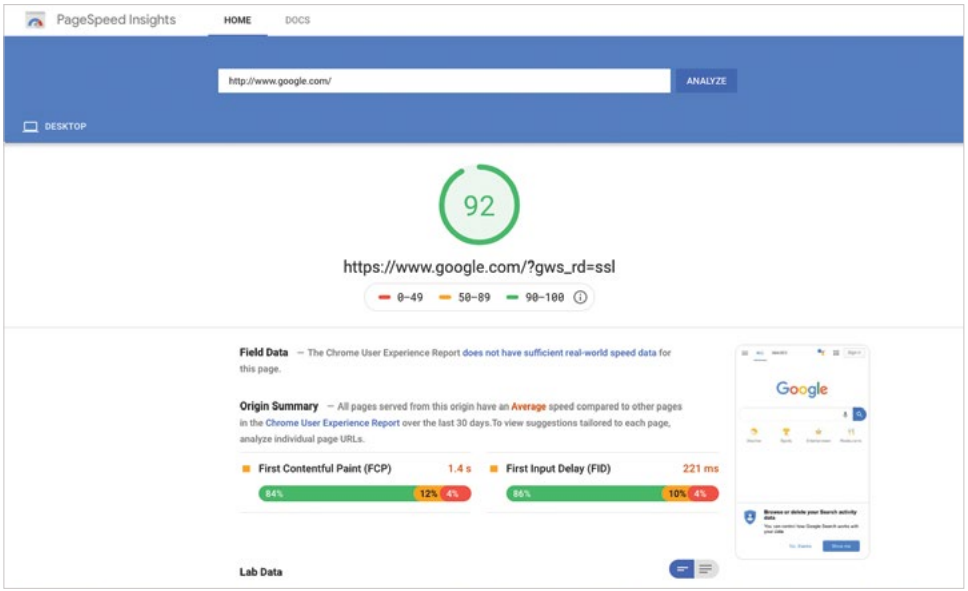
Data issues are usually isolated to a specific page, section, or data set on the site. Sometimes data issues can seem like functional issues if something isn't showing up as expected. The key difference is that data issues can typically be mapped to a specific setting or data entity such as posts, pages, categories, tags, and menus. Functional issues will tend to map to broader aspects of a page such as everything after some seemingly arbitrary point.

Security

Most people only recognize a security issue after their site has been compromised. Any signs of tampering are a clear indicator of a security issues, but strange Google search results for your site could be as well. When dealing with security issues, it is important to determine how the hacker gained access. This could range anywhere from a misconfigured server, a plugin, or simply choosing too short of a password. Failure to address the hacked code or the initial vulnerability will result in subsequent hacks.

Performance

If a page loads slowly or elements on the page behave in a sluggish manner, you are dealing with a performance issue. Unfortunately, there are many reasons why this sluggishness could be occurring. One of the best tools for doing a quick evaluation is to use Google's PageSpeed Insights tool. You can also refer to Section One and address the specific performance issues outlined there.



Environmental

Environmental issues are those that pertain to the services that help run your website, such as the services your web host or domain registrar offers. If an issue occurs when no changes have been made to the site, or when things are broken in a way that you can't otherwise classify, it is almost always environmental.

For example, if a browser says it can't load your site and you get a **500 – Internal Error** message or see some official-sounding message on your screen, you are probably experiencing an issue from a service from your web host or domain registrar. If you aren't sure which service might be the problem, start by contacting your web host.

Potential Sources — and Course of Actions

Now that you have documented the symptoms, contextual data, and know more about the type of issue you are dealing with, you can begin your analysis to determine the root cause of the problem. Your diagnosis doesn't need to be perfect. You only need enough information so that you can take action. In some cases, you will be able to resolve a problem on your own; other times, you may need to rely on others to resolve a problem for you.

There are several aspects of a website that might be the source of your issue. Each of these potential sources has some simple checks you can run to determine if that is where your problem lies. Here they are in a prioritized, recommended order for troubleshooting, with suggestions for actions you can take to resolve the issues.

PRO TIP

A quick reminder: Always create a full backup of your site first.

Plugins

There are two potential sources for plugin issues: versioning and plugin incompatibility.

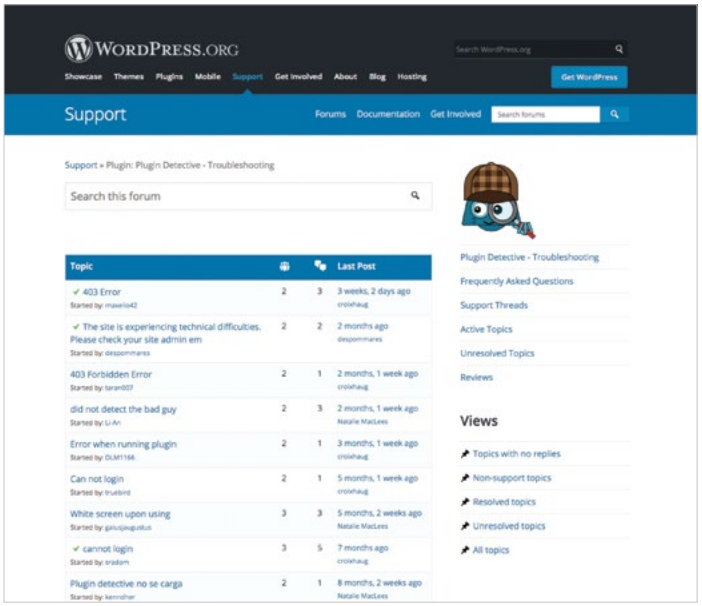
Versioning

One of the first reasons that a plugin may be a source of an issue is that you may not have the latest version installed. Bugs happen, and when they do, plugin authors release updates. If you aren't running the latest version, you may be encountering issues or potentially running vulnerable code that might result in your site being hacked.

If you are already running the latest version of the plugin and still have issues, you should reach out to the plugin author. For paid plugins, this is typically very easy; the company or person behind the plugin will almost always have a way to reach out for support. If you are using a free plugin, how or where you can gain access to support may be unclear. In fact, some authors of free plugins don't offer support of any kind. Keep in mind, if you do reach out for support from someone offering a plugin for free, be respectful and appreciative of the fact that the person is willing to spend time to help you out.

Each plugin listed on the **Plugins** page in the admin dashboard should have a **View Details** or **Visit Plugin Site** link. All plugins hosted on the **WordPress.org Plugin Directory** will have a **View Details** link that will open a modal revealing a **WordPress.org Plugin Page** link. Clicking this will take you to the plugin page on WordPress.org.

From there, you can navigate to the **Support** section and create a new issue at the bottom of the page, assuming you have set up a free account and are logged in.



PRO TIP

Remember, when reaching out for support, provide all the documentation, context, and research you can. This helps provide clarity for the people trying to help you and can cut down on the back-and-forth communication as they try to wrap their heads around your specific scenario.

If you are still having issues with the latest version of the plugin, you may want to simply use an alternative plugin that does the same thing. In many cases, there is more than just one plugin that can get the job done, and it might be worth your time to at least give other options a try. Another benefit of switching plugins is if the one you are currently using has a lot of features that you don't use. It would be better to downsize the amount of code and bloat on your site and find a simpler plugin that does just the one thing you need.

Plugin Incompatibility

It is quite common for two plugins to not to get along with each other. The WordPress plugin ecosystem is very large, and no author can make sure a plugin works in all circumstances and with every other available plugin. To make matters worse, people who write code are human, so they have a tendency to make mistakes. Sometimes those mistakes are released in a new version of a plugin that you just happened to download along with 10 other plugin updates.

The first step in checking if a plugin is an issue is to deactivate all your active plugins. You don't need to delete them, just deactivate them so their code doesn't run.

Not all plugins will directly impact visitors to your website, but you probably have a handful of plugins that, if deactivated, would break some functionality for your site's visitors. For this reason, it is best to use a staging site if you have one available.

After you have deactivated all your plugins, check to see if the problem still exists. If it doesn't, then one of your plugins is responsible. In some cases, you may find that you can't get to the thing you need to test because a plugin added that functionality. If this is the case, the plugin that added that functionality is probably the culprit as well.

Assuming that one of your plugins is indeed responsible for the issue, how do you go about determining exactly which plugin is responsible? As you might have guessed, you need to activate each plugin one at a time to see if the problem returns. If it does, the last plugin you activated is the culprit.

There is a way you can shortcut this process a bit. You can bisect your plugin list and reactivate the first half of all your plugins. If the problem returns, one plugin in that subset is responsible. If the problem doesn't return, a plugin in the other half of the list is responsible. Either way, you just reduced your work to find the problem plugin by half. Repeat this process as many times as necessary by halving the remaining pool of potential problem plugins and you will be surprised how quickly you will find the troublesome plugin.

If this process is making your head spin a bit, don't worry. There is a plugin called Plugin Detective which will automate this process so you don't have to think so much.

One thing to keep in mind when dealing with plugin conflicts is that it takes two plugins to create an issue. In other words, it is entirely possible that disabling either one of two conflicting plugins could resolve the issue.

For example, let's assume that when you disable two active plugins both, the issue is fixed. As you toggle these two plugins on and off to find the culprit, you may find that when you disable the first plugin, the problem went away. However, this might be entirely by chance. Depending on the order in which you toggled the plugins, it may be that the second plugin was doing something incorrectly and happened to conflict with and break the functionality of the first one.

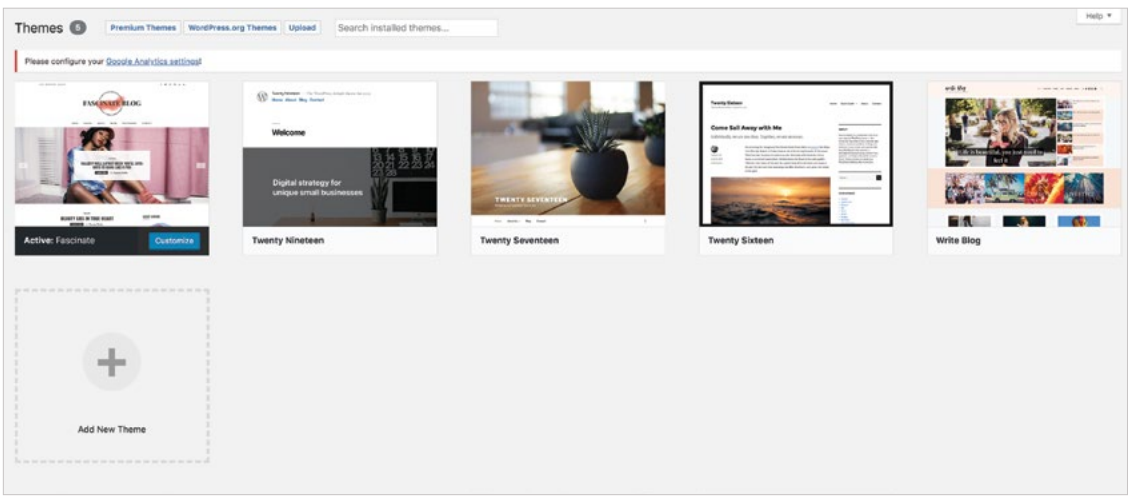
In order to be certain that you haven't mistakenly assumed that only one plugin is responsible, you can repeat the plugin toggling process while keeping the problem plugin active at all times. If at any time the problem goes away when a particular plugin is inactive, you've found the other incompatible plugin.

Theme

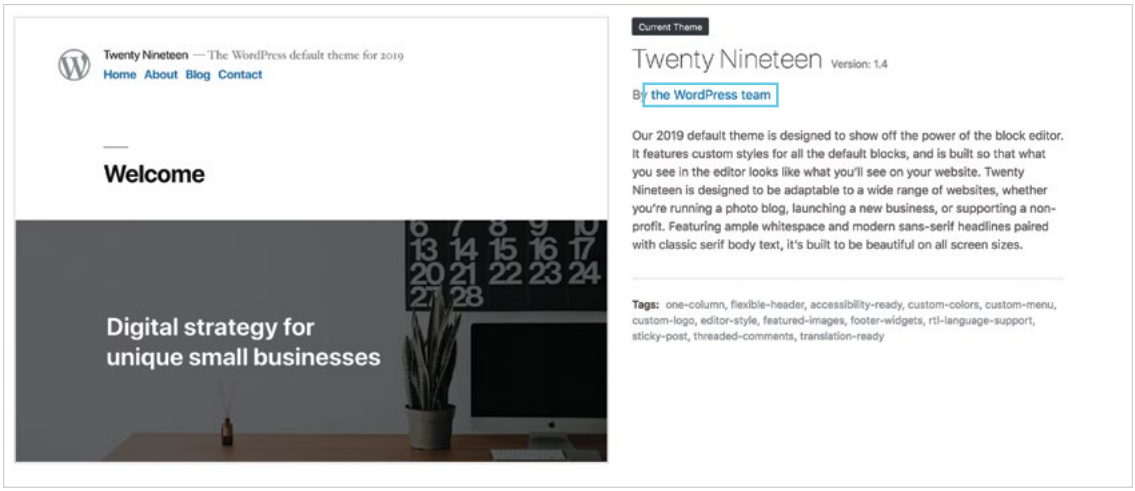
Another common potential source of issues is your theme. You can only run one theme at a time, so if your theme is the issue, it would have to be your currently active theme. Start by making sure you are running the latest version.

If you are, you can try switching themes to see if that resolves the issue. Before you do go down this path, we once again recommend that you back up your website or, if you have one, use a staging site before switching themes — just in case something goes wrong. A staging site is preferred since switching your theme on your live site will be visible to your site's visitors. However, you will need to make sure your staging website has been synchronized with your live website before you start.

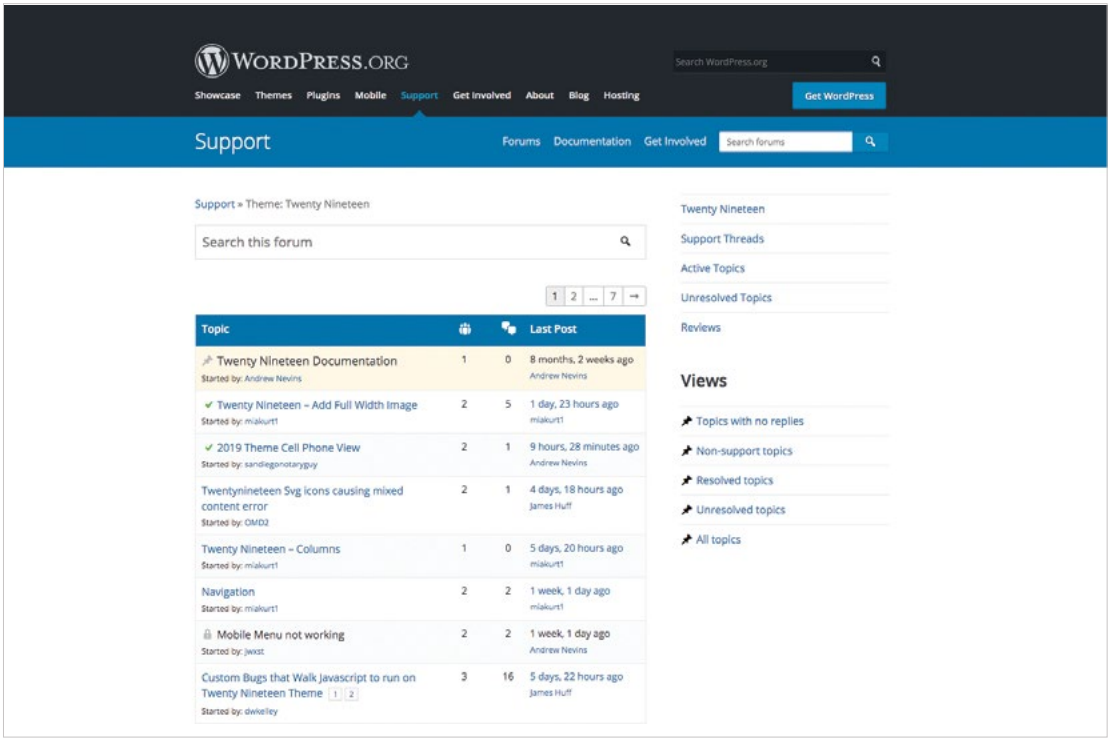
You can switch to a default WordPress theme such as Twenty Nineteen, or other alternative themes. To find theme options, go to the **Appearance** page in the admin dashboard to see a list of all your themes. Hovering over a theme image will reveal a **Theme Details** link that you can click.



The next screen typically has a link to the theme's website.



If you are still having issues and think it's your theme, you can reach out for support. If you downloaded a theme from the WordPress theme directory, the best way to reach out for support is to visit *wordpress.org/themes*, search your theme by name, then locate the support section on the resulting theme page.



If the problem persists after you've switched out themes, and you've already checked your plugins, then check the other potential sources of issues.

Caching

Caching is the act of temporarily storing a resource so future requests can be served faster. For instance, browsers store important files so that subsequent pages load faster. Web servers can temporarily store copies of generated files in order to avoid the overhead of running the same code on every page load. Domain name servers (DNS) cache domain to IP address mapping so that visiting a URL doesn't require constantly scouring the web to find the right computer.

In general, caching is a beautiful thing. It can help make your website lightning fast. In reality, it can also cause lots of trouble. If you misconfigure a setting in a caching plugin, you may end up serving old data to users for too long. If your cache hasn't expired yet and you make a change on your site, you may not see that change on the front end right away. If you run multiple caching plugins, you are bound to run into odd issues that are hard to pinpoint.

Understanding the basic concept of caching as well as the contextual nature of it will help you better diagnose a caching issue. As mentioned earlier, browsers will store files on a person's computer to help speed up web page loads. Typically, they only do that if they are told to. It could be a plugin, web host, or some intermediary layer that tells the browser to cache a file. Fortunately, browser caching issues can easily be tested. Just open up the problem webpage in a different browser or computer. If you use Chrome, you can also just open up an incognito window. If the problem only exists on your browser or computer, it is likely a caching issue.

Often web hosts or other service providers like Cloudflare will automatically enable caching. In some cases, people are entirely oblivious to the fact that caching is happening. Before trying to use a WordPress caching plugin, you should contact your web host and perhaps any other service providers to see if they have implemented caching on your behalf. If so, they should have some form of web interface where you can go in and disable it or empty all cached content for testing purposes. If you think you have a caching issue, make sure you turn off or flush any cache you are aware of. If the problem goes away after doing so, then you had a caching issue. Remember, just because you have a caching issue doesn't mean you should completely disable caching. You just need to dig deeper to understand what went wrong and how you can avoid that in the future.

Another common approach to caching is to use a WordPress plugin. If you followed the troubleshooting process for plugins, you may have identified a caching plugin as the culprit. Of course, it isn't the plugin that is the problem. The real issue is the configuration of the plugin or if you are trying to use more than one caching plugin at the same time. Again, you may need to do some research and exploration to find out what went wrong and how to avoid it moving forward, but at least you know where the problem originated.

You may notice that when you are logged into your website you can see all your changes, but when you are logged out, it looks as though nothing has changed. Almost without exception, this is due to a WordPress caching plugin that excludes logged-in users from being served cached content. What this means is that those who are able to login to the back end and edit content will be able to see their changes. Meanwhile, visitors to the front of the site who aren't logged in will continue to see the old content until the cache expires; typically within one to a few hours. While this is actually desirable in most cases, it points to another simple test you can do to see if you are dealing with a caching issue: test a page from both the logged in and logged out states. If there is a difference between the two, you probably have a caching issue.

Caching can be split into two primary types of issues: temporary and recurring.

TEMPORARY ISSUES

Temporary issues typically come about when you expect the cache to be cleared and to serve up the most recent data or changes, but it doesn't. Once the cache is cleared, it will serve the latest updates and the problem will be solved. This is just a matter of determining what caching layer is responsible for the issue and then flushing the cache. Start by flushing your browser's cache, then check if your host does caching and flush that, and finally, check if a WordPress caching plugin is installed and flush that. If you happen to be using a third-party tool like Cloudflare, you may need to flush the caching there as well.

RECURRING ISSUES

Reccuring issues are persistent issues that may go away on their own but happen continually. These issues can be very annoying and potentially detrimental to your site or business if not dealt with promptly. Below is a list of three common causes of recurring caching issues and how to handle them.

Misconfigured Settings

It can take a lot of know-how to configure a caching plugin perfectly. However, most issues arise due to one of two reasons.

SOMETHING GETS "TOO OPTIMIZED"

This is a common issue with optimizing JavaScript, for instance. If the code gets rearranged, it can often result in mangled execution logic and break interactivity on web pages.

CONFIGURATION DOESN'T TAKE INTO ACCOUNT DYNAMIC PAGES

For example, if you have an eCommerce site with a shopping cart, there could be a situation where one visitor fills up a shopping cart and then that page gets cached. All other visitors trying to view their shopping carts would see the same shopping cart, which isn't what they wanted to order. Or, consider a page containing a nonce (a text string used for security purposes to validate a user's intent to perform an action) gets cached. Then an important form that relies on it, such as an eCommerce checkout form, can't validate correctly and stops working. Now there is a caching issue that will cost you money!

Conflicting Layers

People often have the mentality that if caching is good, more caching must be better. While there is some truth to that, it only applies when you have multiple *different* caching layers. For example, it is fine to have object caching, page caching, and browser caching all working together. However, when you have two or more of the *same* kind of caching layers, you will run into trouble. For example, if you have two WordPress plugins trying to do the same caching, you've more than doubled the chance that you will run into strange issues on your site.

Cache Busting Not in Use

Cache busting is the act of automatically expiring the cache when a particular event, typically an update, occurs. This ensures that the cached data is never outdated. For example, every time you update a post in WordPress, it should trigger the cache for that specific post to be busted. Likewise, if you are making changes to your theme's styles, if you don't update the version when enqueuing the file, it won't bust the cache and it will appear that nothing has changed.

Ultimately, it will be up to you to apply your critical thinking skills when dealing with caching issues of any kind. Sorry, there is no "easy" button on this one, unless you just opt to use a paid caching service or plugin where you can just call support when you have a problem. Of course, if your web host handles caching and that is the problem, just contact your web host's support team.

Data

Issues with data can be a bit tricky. There is no easy way to effectively rule out a problem with data. However, by applying some basic logic, you can reasonably estimate the likelihood of bad data being the culprit.

At this point, the assumption is that you have already ruled out about 90% of all code-related issues on the site because most issues are going to be plugin- or theme-related. The likelihood of an issue being data related at this point is relatively high. Keep in mind that data issues aren't always directly related to your site content. More often, configuration options for your theme or a plugin are to blame. So, if you've already identified your theme or a plugin as an issue in a previous step, you may want to look closer and see if what you are really dealing with is simply a misconfiguration.

The nice thing about data issues is that, as long as you don't change the data, the issue will be very consistent. At the same time, if you suspect that specific data or settings are responsible, you can just edit or delete the data and see if the issue persists, changes form, or goes away.

Additionally, because data is contextually relevant, you can also reliably assume that your issue will be isolated to a specific page, section, or data set on the site. If you are able to detect a pattern and correlate that to a specific configuration setting, taxonomy, post type, menu, or other such data entity in WordPress, you can be pretty certain you are dealing with a data issue. It is also possible that a data issue only affects a single page. In such cases, make sure the issue is consistent. Any inconsistencies mean you are not dealing with a data problem.

If you've identified a data-related issue, it is typically self-evident as to how to fix it. However, there may be a few situations where that is not the case. For example, if there is a data input in a plugin or theme that results in strange side effects when changed, it might be worth reaching out to the plugin or theme author (see the discussion on plugins or themes accordingly). In some cases, simply re-entering data a different way, such as manually typing it out instead of copying and pasting, is enough to do the trick.

Less common, but extremely frustrating, are character encoding issues. When character encoding gets mismatched, it will result in strange characters, also known as "gremlins" showing up throughout your content. If this occurs, you will need to meticulously go through your browser settings, HTML, HTTP headers, and database to make sure they all use the standard UTF-8 character set.

Third-Party Integration

In some cases, you will run into an issue where the integration of a third-party service fails and causes outages and technical complications. For instance, this could happen if you use a third-party service for tweets on a web page, comment boxes on blog posts, or web forms.

If any one of these services is down, it could cause content not to show up or even break functionality on your website. When these services go down, the best course of action is to check and see if the service in question has a status page. Many services like Twitter, Facebook, Instagram, Salesforce, Hub Spot, Amazon, Cloudflare, and many others have standalone pages that simply monitor and report the status of their services. These are usually easy to find by Googling the company name with the words "Status Page" appended.

If the status page shows that the service has only been down for a short period of time, the best course of action is to wait for a bit and see if it comes back up. However, if the service has been down for an extended period of time and there is no indication on the status page that the problem is being worked on, you may want to contact the company directly and inquire about the issue.

However, if the status page shows that the service is working fine, the problem could lie with an integration plugin that you are using. If that is the case, take a look at the appropriate course of action for dealing with plugin issues.

Server

As a general rule, if you've checked all the other potential sources of issues, and came up empty, it is probably related to the server environment.

Servers are typically owned or leased out by your web host and located somewhere in a large facility alongside thousands of other servers. These computers have web server software like Apache or Nginx installed. The software is responsible for making sure that requests for a web page are properly routed to your website. As such, when using the term "server," we are actually referring to not just the configuration and performance of the computer, but also the configuration and performance of the software on the machine.

Web hosts typically provide an interface for managing certain but not all aspects of the configuration. This includes things like the PHP version, the state of the WordPress database, memory and execution limits, and the configuration of the web server software itself. If you know what you are doing, you can address certain issues on your own.

Certain server issues may have been caused by changes you've made to configuration files on your site or by settings you changed in your web host's administrative dashboard. If you or someone else with access to your site made these types of changes, the next step would be to revert those changes to see if that fixes your issue. Of course, you may need to reinstate your changes later, with appropriate alterations to make them work, if your site needs them to function correctly.

If you haven't made any changes to configuration files or in your host's dashboard, take a look at your PHP error logs. Sometimes code changes can introduce errors that will bring a site down.

It isn't uncommon that you may need to contact your web host to get help with some of these issues. And generally, if your website isn't loading at all, reaching out to your web host to see what is going on is a good idea. It could be that your server is temporarily down, a misconfiguration occurred on the web server, or some bad code is keeping your site from loading. In all these cases, your web host should be able to help you navigate what is going on and your options.

WordPress

At the core of everything, there is WordPress. It is uncommon to run into an issue that is caused by WordPress itself and you need to be able to rule it out. The best way to validate whether the problem lies with WordPress or not is to install a clean copy of WordPress somewhere and see if you can replicate the issue without installing any themes or plugins. Since WordPress is most likely not the issue and this test does require another install of WordPress, it is the last thing you should check.

If you've identified an issue as a bug in WordPress itself, this is your chance to improve the software for millions of people. Visit the WordPress website and follow the steps in the handbook for reporting a bug.

Finally, if none of the above applies or you've already completed each of these checks, contact your web host's support team.

Common Issues

The best way to sharpen your troubleshooting skills is to jump in and start. But much like how you need to use a GPS to find a new location for the first time, it's good to have a little help the first time around. The goal for this section is to introduce a few common issues and how you can use the recommendations here to work through the problem. These examples will assume that documentation and a site backup have already been completed.

Issue #1: Email Deliverability

SCENARIO

You just set up a new real estate website that allows people to login, save properties they are interested in, and get emails about new properties that meet their criteria. The only problem is, nobody is getting any emails.

ISSUE TYPE

You work your way through the list and easily rule out visual, interactive, security, and performance as issue types. You aren't certain if this could be a functional, data, or environmental issue. Since everything loads fine on the site, functional doesn't seem to be a good match. On the other hand, something isn't happening that should, which seems to indicate a functional issue. Data is a possibility since it is related directly to the email content you want to send out, but that seems unlikely. Environmental seems like a good fit because email is a supporting service.

SOURCE

Disabling the plugin responsible for the real estate functionality disables the features you want to test. It is possible that the plugin is the issue.

On the other hand, your email is a separate service from your web hosting, so maybe it is a third-party integration issue.

Just for good measure, you decide not to rule out the server since there aren't any good tests to prove that this is an issue or not.

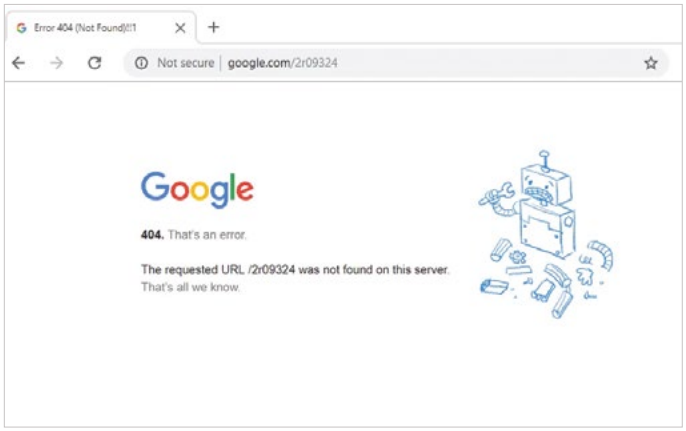
COURSE OF ACTION

The plugin has 24/7 support, so you make a quick call and ask why emails aren't going out. The company says it doesn't offer support related to email issues and tells you to call your web host.

You call your web host and explain the issue. The support person there points out that for security reasons, the company blocks all outgoing email sent directly from the server and you should look into using an SMTP plugin or transactional email service.

You do some research and find out that when using a WordPress simple mail transfer protocol (SMTP) plugin, you can plug in some information about your client's Gmail account and have all the emails go through Gmail. Unfortunately, due to the scale at which emails go out, you learn that many of your client's emails still end up in people's spam folders.

You decide to look into transactional email services and find one that has a WordPress plugin that makes setup a breeze. Additionally, the service allows you to login to a dashboard where you can see each email in the queue and the current delivery status. Nearly all your client's emails are now sending without a problem and the couple that don't can easily be resent.



Issue #2: Page Not Found

SCENARIO

You just installed a cool new plugin that helps manage business listings. It creates a new post type called **Listings** and lets your client manage each listing separately. You've already entered 10 listings just to give the plugin a trial run. You can see all the listings in the admin dashboard but once you click on the link to view a listing on the front end of the site, all you get is a 404 error page. All the other pages on the website work just fine, except those individual listing pages.

ISSUE TYPE

This is what appears to be a classic data issue. It is a consistent problem that only affects a specific section of the site and directly correlates with a post type in WordPress. At the same time, you don't want to be close-minded about this, so you take into consideration that it could be a functional issue as well. After all, you're pretty sure the plugin isn't working correctly.

SOURCE

After running your tests, your problem goes away when you disable the listings plugin, but so does your desired functionality. You determine that the plugin is the source.

COURSE OF ACTION

First, make sure you are running the latest version of the plugin. Once you determine you are, reach out to the plugin developer for support and provide all the details of your issue. The developer gets back to you a day and a half later. She wants you to visit the permalinks page in the admin dashboard and then try and load up one of the individual listing pages. Magically, a simple visit to a particular page fixes the issue. The developer explains that WordPress stores rewrite rules in the database and her plugin wasn't properly adding in the new rules when the plugin was activated. She updates her plugin and pushes out a new version. It turns out it was both a functional issue with the plugin and a data issue!

Issue #3: 500 Internal Server Error

SCENARIO

Every page on your website is showing up as a 500 error page.

ISSUE TYPE

As you know, functional issues occur when a page doesn't load as expected. However, this could also be a server related issue because your site isn't loading at all. Either one of these options could be correct. It depends on the actual problem.

SOURCE

When trying to identify the potential source, you can't run through the tests for plugins, themes, caching, data, third-party integration, or WordPress. After all, your site isn't actually loading. The only remaining possible source is the server.

COURSE OF ACTION

The first step is to see if you've made any changes to configuration files. Let's assume for a second that you did. You've updated an **.htaccess** file to add a new security rule you copied from a blog post.

After undoing that change, you find that the site loads again. You do a bit more research and find that the blog post you copied it from was using an invalid code sample and you just needed to make a small tweak to make it work. It turns out the issue was an environmental one.

Let's pretend you didn't make any changes to configuration files. The next step is to double-check the error logs. In this scenario, you find that there is a fatal PHP error that points to a specific file and line of code on the server. As the developer, you remember you just made a change to that file. A few minutes later, the site is back up. In this case, the issue was a functional one.

What if all those things didn't turn up any clues? You contact your web host and share all the information you have about what is happening. The support tech digs in and realizes that something happened and some file permissions were incorrectly set. She corrects the issue and the site is back up before you end the call.

Becoming an Expert Troubleshooter

Hopefully, you can see the value of having a mental framework to help you triage and handle issues. You've learned how to properly document an issue and how to leverage additional context to aid in your diagnosis. You know how to classify issues and narrow down potential sources in order to form a clear course of action. You've also seen a few examples of the framework in action.

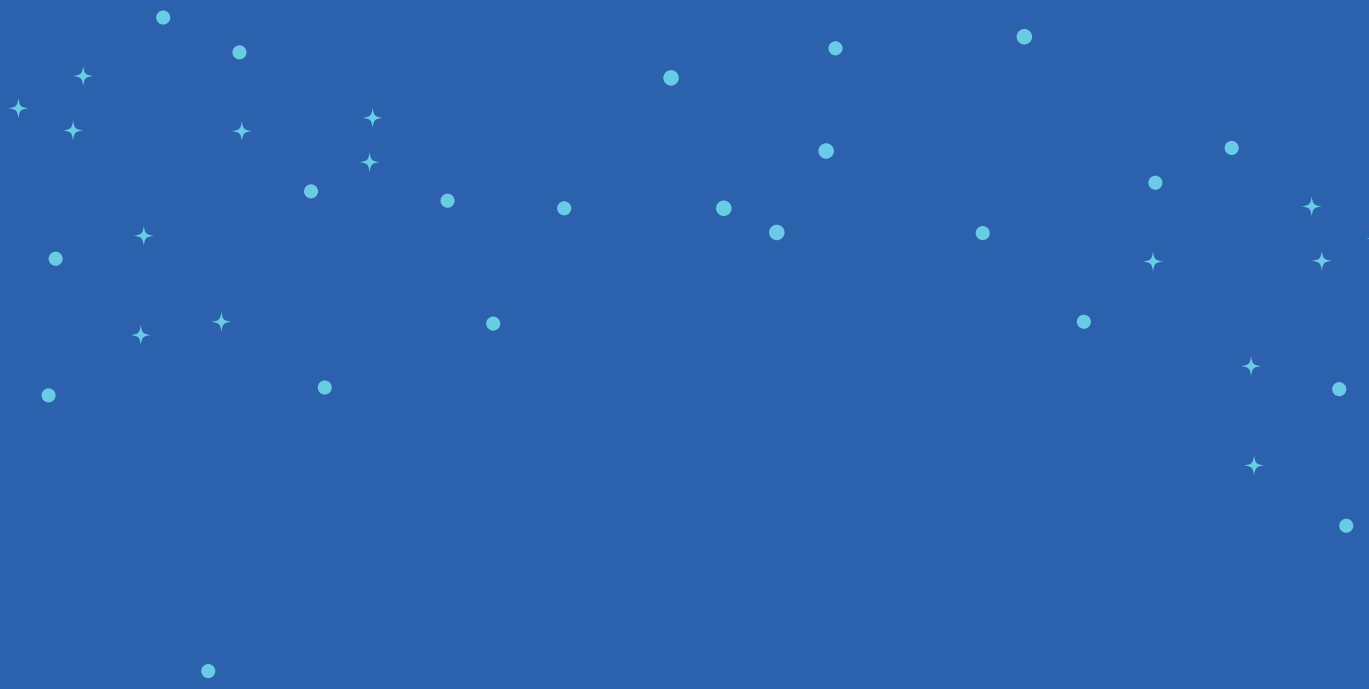
As you can see, troubleshooting isn't always straightforward. Sometimes you have to back up, challenge your assumptions, and reframe your hypothesis. Keep trying things and ruling out possibilities.

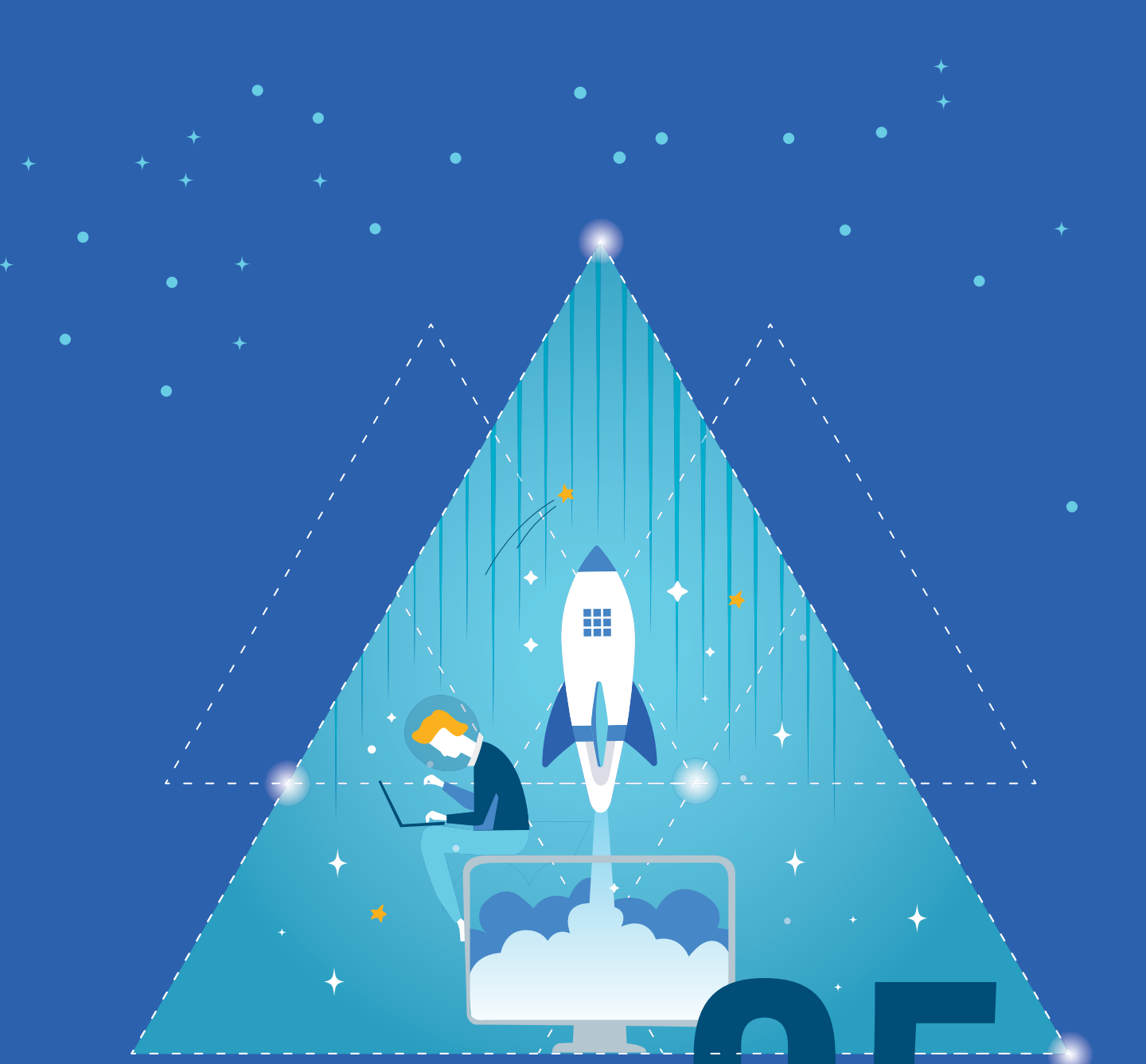
> Don't be afraid to ask for help. Every problem you encounter is an opportunity to expand your knowledge and experience. Now go and practice your new superpower!

SECTION TAKEAWAYS

In summary, here is a checklist you can use to begin troubleshooting issues.

- ○ Identify and document issues. Include the essential and contextual details.
- ○ Figure out a workaround until the issue is resolved.
- ○ Diagnose issues by using a scientific approach. Start with a hypothesis, identify appropriate tests, and start testing.
- ○ Use your diagnosis to analyze the root of the issue.





SECTION FIVE

MIGRATIONS

05

MIGRATING WITH EASE

There will be instances where you want to migrate, or move, one of your WordPress sites from one web host or server to another. This can seem like a daunting task, and to many people, it is just that.

In this section, we are going to show you two different ways of migrating a WordPress website: manual or plugin-aided migration. This should hopefully give you the confidence to migrate with ease. Before you start, however, you will need to make sure you set up a couple of things.

Access

The first thing you need to do is to make sure you have the required access for accomplishing this task, whether you are completing the migration for your website, or for someone else's.

ADMINISTRATOR ACCESS

If it is your website, you will most likely already have this, but if not, you need to get it from the website administrator. If you are moving a website for a client, you will need to request that you are set up in the administrator role or super admin role for a multisite.

OLD AND NEW HOSTING ACCOUNTS ACCESS

You will also need full access to both the old and new hosting accounts.

FTP ACCESS

If you have access to the old and new hosting accounts, then you can set up File Transfer Protocol (FTP) access. FTP is not required if you have control panel access to both hosting accounts, however, it will make it easier for you to download and upload the files from one server to another. For security purposes, you should always use SFTP.

Tools

You will also need may need some tools, depending on which option you choose.

AN FTP PROGRAM

FileZilla is free to download and will work on Windows, Mac, and Linux, but you can use any FTP program.

MIGRATION PLUGIN

This is not required for the manual migration process. It will, however, be required for the automated migration process that uses plugins. There are many free and premium plugins available to help you accomplish a somewhat automated migration, but for our tutorial, we will be using the **All-in-One WP Migration** plugin, which is one of the most popular options. It is free to install from the WordPress.org plugin repository.

Migration Options

Before we get into the two migration options, let's talk about the pros and cons of each approach.

The manual migration process gives you better control but is more labor intensive and requires a much higher level of technical knowledge to complete. Plugin-aided migration is an easier process that can be completed by a more novice user, but it may not work on larger sites and you will have less control over the process. By the end of this section, you can decide which method will work best for you and your website.

Manual Migration

PROS

- More control over the process
- Ability to move any size site

CONS

- More labor intensive process
- Higher technical knowledge required

Plugin-Aided Migration

PROS

- Easier process
- Less technical knowledge required

CONS

- May not be able to move large sites
- Less control over the process

Manual Migration

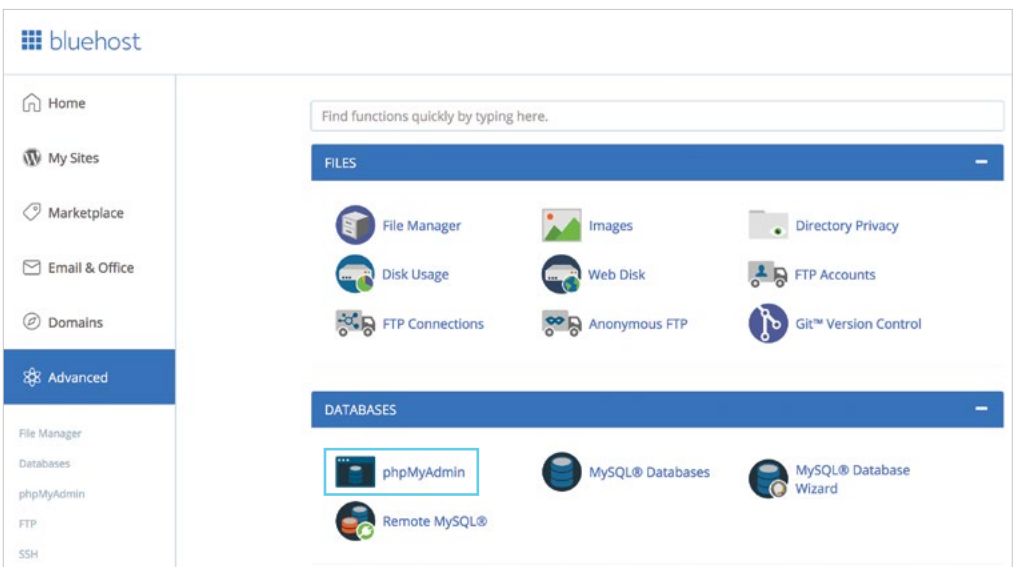
Before you begin, you should make sure that WordPress Core and all themes and plugins are updated to their latest versions. Additionally, you'll will also want to back up your site as well. Once those tasks are done, follow these steps:

STEP 1

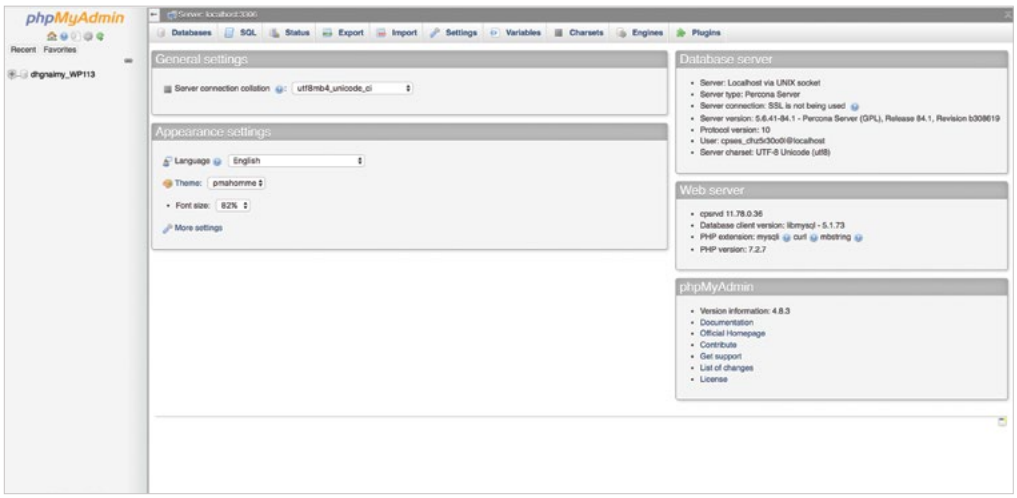
Log into the Current Hosting Account

The first step in the manual migration process is to log into your control panel or other graphical user interface (GUI) of the current hosting account. Once inside, locate **phpMyAdmin,** a free software tool that helps you manage the administration of MySQL. This can be found in the Bluehost control panel under the **Advanced** tab.

Next, log in by clicking on the **phpMyAdmin** link. If you can't log in that way, then you will need to enter the database username and password into the login screen. To find these, look at the `wp-config.php` file for the site, which can be found in the root directory of your WordPress installation. You can download the `wp-config.php` file using your FTP program or open it in your browser using the **File Manager**. You will need the information from this file moving forward, so it's good to download it now.



Once you are logged into **phpMyAdmin**, you should see this screen.



STEP 2

Find the Database

Your databases will be listed on the left side of your screen. Select the database that you would like to export to the new host. If your database contains only one WordPress installation, select it and move on to the next step. If there are more, like on the screen above, determine which database you need to export through one of these steps:

- Open the `wp-config.php` file that you downloaded earlier and look for the `DB_NAME` constant. It will generally be the same as the `DB_USER` constant.

- Alternatively, click on a database from the list, then click on the `*_options` table to open it. The first two items in the `*_options` table should be **siteurl** and **home**. Each option contains the URL of the website this database is associated with. Look through each table until you find the correct database. Note: The asterisks above will be replaced by your specific table prefix. Many times the table prefix will be `wp_` but not always. You can find your table prefix in your `wp-config.php` file.

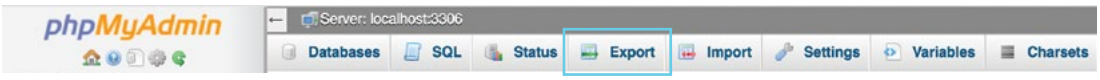
Once you find the right database, click into it so you can see all the tables listed.

STEP 3

Export the Database

Now it's time to export the complete database for this WordPress installation. Click on the **Export** tab in the top toolbar.

Leave the **Quick** option selected and click on the **Go** button. This may take a little time, depending on how big your database is. You'll end up with a `.sql` file which you will need when you import your site into the database on your new hosting. The file should be named `YOUR_DATABASE_NAME.sql`. Make sure to save this in a place where you can find it quickly. Close **phpMyAdmin** and move on to the next step.



STEP 4

Get the Files

Now you need to download all the necessary files. You *do not* need to download the complete WordPress installation. All you really need is the `wp-content` directory, which contains all the themes, plugins, and media uploads from your website. The core WordPress files will already be on your new server once you have created the new install. You should always check to make sure that the WordPress Core versions match in both your old site and your new site. Mismatched versions could lead to issues during the migration.

This is where an FTP program will come in handy. Open your FTP program, connect to the current server, and download the complete `wp-content` directory to your computer. This will most likely take a while. Grab a coffee or lunch and come back when the download is complete.

That does it for the current old site. Now it is time to move on to your new hosting account.

PRO TIP

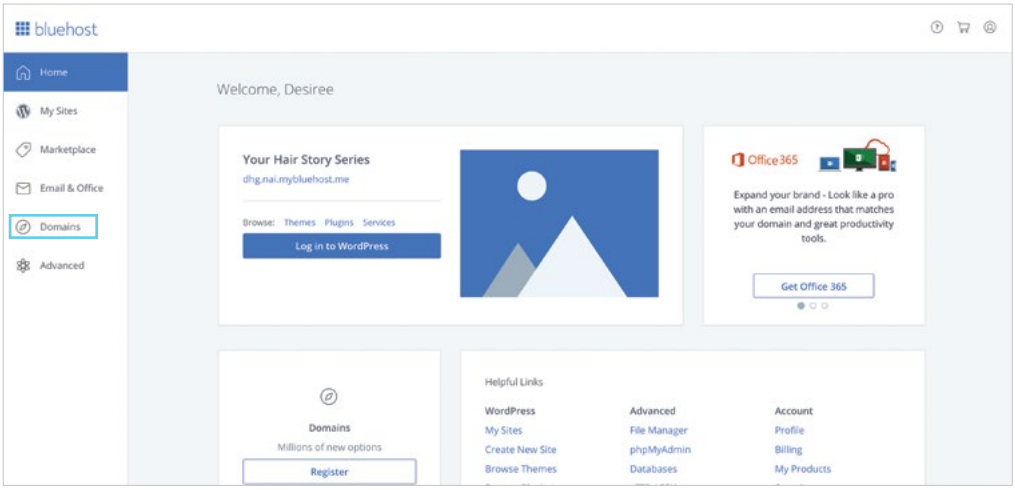
Log into the control panel and go to **File Manager**. Here, you can compress the entire `wp-content` directory before downloading it via FTP. This will compress the directory to a much smaller sized .zip folder. Make sure to uncompress the zip folder after uploading it to the new server.

STEP 5

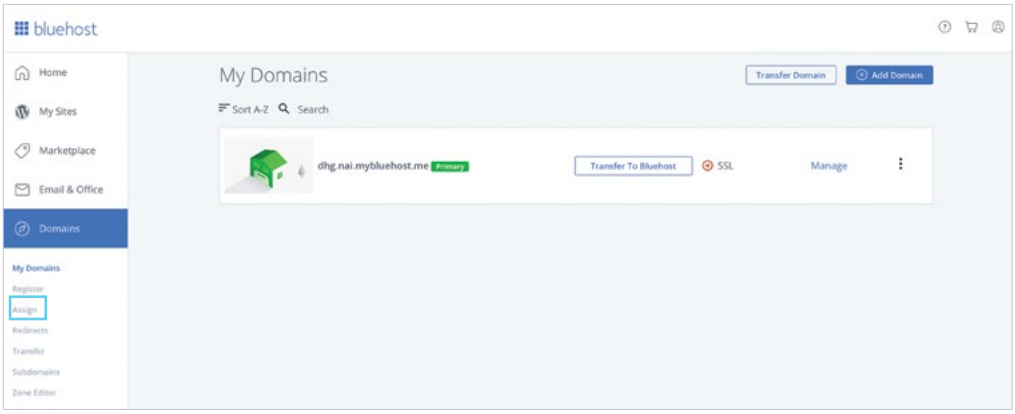
Add your Domain

In order to set up the new WordPress install, you first need to add your domain to your new hosting account as an **Add-on Domain**. Though the next set of steps and screenshots are specific to Bluehost, most of the steps will apply to any hosting account used.

Log into your new hosting account at *https://my.bluehost.com*. This is the screen you will see.



Click on the **Domains** tab, then click **Assign** from the left sidebar menu.
This will take you to the **Assign Domain** page where you can assign your domain.



Add your domain to the account and verify that you own it through one of the following options.

- Use an EPP authorization code which can be obtained from your current registrar. This is a good option since it gives you time to complete the migration before switching the DNS to point at the new server.
- Some hosts will also allow you to add an HTML page to your current server with a specific code for verification. This is also a good option since you already have access to the current hosting account.
- Point your **nameservers** to your hosting account.

You can also transfer your domain to the hosting company you are using if it offers domain registration. Bluehost makes this very easy and it is a free service.

Once your domain is verified or has been transferred, you can move on to setting up the new WordPress install within your new hosting account. It can take a bit of time for domains to be verified or transferred, so keep this in mind.

PRO TIP

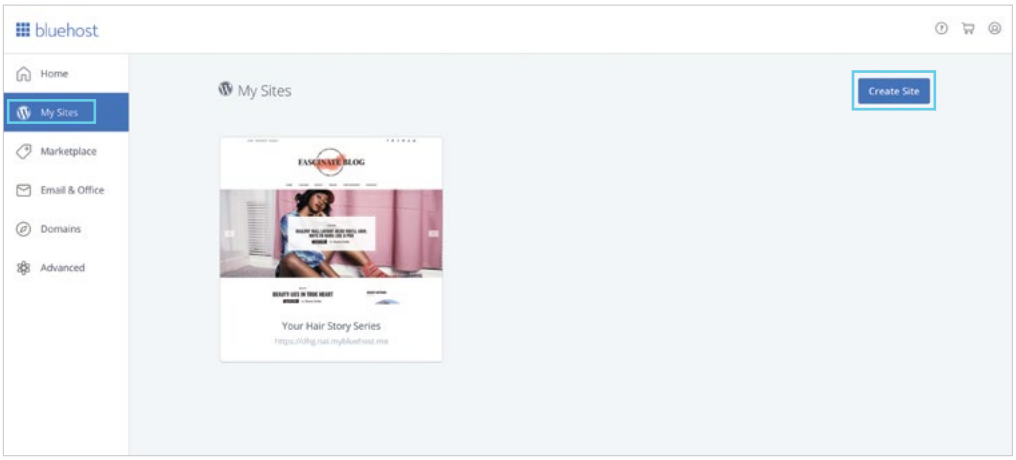
Do not point your **nameservers** to your new hosting account until the migration is complete. Doing this could lead to a blank or broken site.

STEP 6

Setup the New WordPress Installation

Most hosts have a one-click install for WordPress. For the purposes of this tutorial, we will once again use Bluehost as an example, but these steps should work for most hosting environments.

Click on the **My Sites** button and then click on the **Create Site** button to set up the WordPress install that you will use to import your site to the new hosting site. Follow the instructions to create your new website, which may vary by provider.



Once your new WordPress install is set up, you should receive a welcome email at the email address specified during the install. For Bluehost users, this email will be delivered to the primary email address of the account holder. As a note; you will be replacing the entire database of this installation, which will include the user info listed in this email. Any logins created during the installation process will not work after the database is replaced due to your domain still pointing at your old server. If you try to visit the new site at this time, you will be sent to the site on your old server.

STEP 7

Replace the Database and Files

Now it is time to upload the files you downloaded earlier, including the `wp-content` folder.

Connect via FTP to your new server and find the root directory of your new WordPress install. Then download a copy of the `wp-config.php` file from your new install. Be careful not to overwrite the `wp-config.php` from your old server, as you may need some of the information from it. The only thing you need to change in the new `wp-config.php` file is the `$table_prefix` variable. This should be changed to match the `$table_prefix` from the old `wp-config.php` file and saved. Now upload this file back to your new server in the root directory. Overwrite the current `wp-config.php`.

Now log in to **phpMyAdmin** on your new server and locate the database for the new install. If there is more than one database, you may need to refer back to the `wp-config.php` file in order to find it. Click into the database, select all the tables, and drop them. You may want to export a copy of this database before completely dropping the tables, just in case something goes wrong during the import of the database from your old website. Once the database is empty, you can now go to the **Import** tab, click **Choose File**, and select the `.sql` file that you downloaded from your old server. This may take a little time, depending on how large your database is.

Next, you can upload the `wp-content` directory into the root directory of your new WordPress install. You can either overwrite the current `wp-content` directory or delete it completely before uploading. *Do not* touch any of the other files or directories.

STEP 8

Edit your Host's File and Test, Test, Test

Since your domain is not yet pointed at the new server, you will need to edit your host's file in order to view the website located on the new server from your local machine. You will need to know the IP address of your new server in order to do this, which you can find on your control panel. Editing your host's file is a fairly easy process that is outlined below.

For Mac Users:

1. Open a **Finder** window.
2. Select **Applications** from the sidebar.
3. Double-click on **Utilities**
4. Double-click on **Terminal**
5. Type `sudo nano /etc/hosts` and then hit return.
6. Enter your administrator password and then hit return.
7. Add the new server IP and the domain, similar to this: **0.0.0.0 www.yourdomain.com**
8. Once you're done, hold down the **control** and **O** keys to save the file, then **control** and **X** to exit.
9. Back in the Terminal, type `sudo killall -HUP mDNSResponder` and then hit return.

For Windows 10 and 8 Users:

1. Press the **Windows** key.
2. Type `Notepad` in the search field.
3. In the search results, right-click **Notepad** and select **Run as administrator**
4. From **Notepad**, open the following file: **c:\Windows\System32\Drivers\etc\hosts**
5. Add the new server IP and the domain, similar to this: **0.0.0.0 www.yourdomain.com**
6. Click **File** > **Save** to save your changes.

Now you should be able to see the website on the new server from your local machine. It's now time to test everything and make sure that all the pages and images migrated over to the new server.

PRO TIP

Make sure to remove the entries from your host's file once you are finished testing.

STEP 9

Switch your DNS

After you have removed the entries from your host's file, you can now switch the DNS for your domain and **go live**. DNS propagation can take up to 24 hours, so make sure to give it some time to propagate. You can use a free tool such as *https://www.whatsmydns.net/* to track the DNS propagation.

Before editing anything on your website, you should check that your **nameservers** have switched. Otherwise, the changes you make will not be reflected once the DNS does switch over.

Plugin-Aided Migration

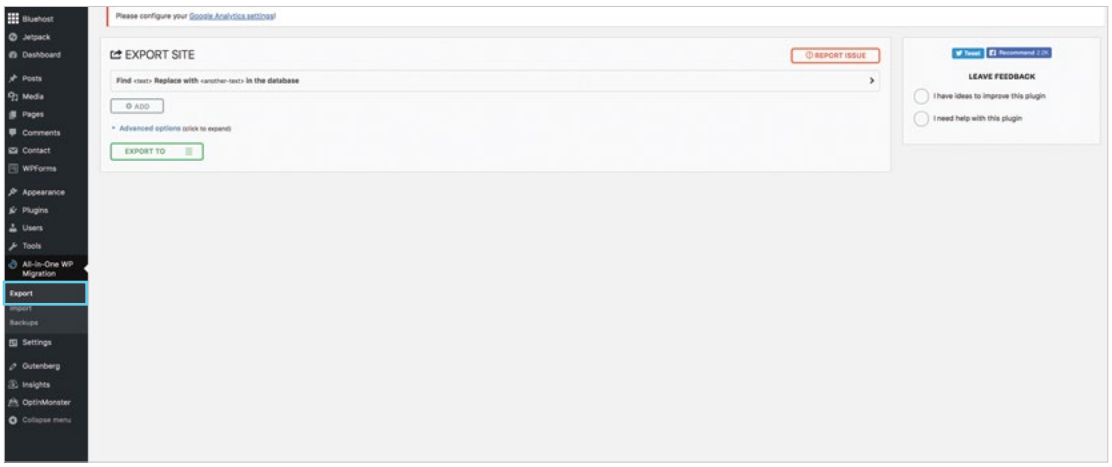
Just like in the manual migration process, before you begin, you should make sure that WordPress core and all themes and plugins are updated to their latest versions. Also, backup your site.

STEP 1

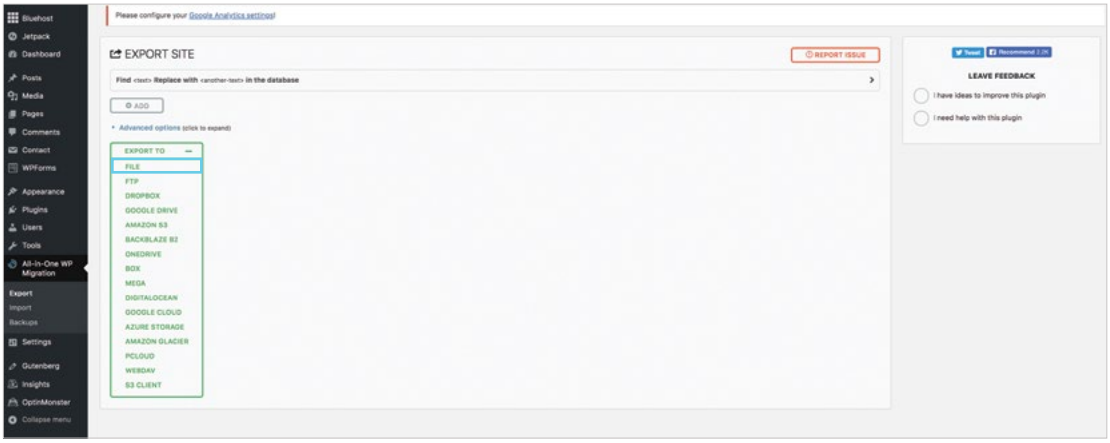
Install the Plugin and Export

Once you're logged in to the WP admin dashboard, install and activate the **All-in-One WP Migration** plugin. Then visit the **Export** page under the **All-in-One WP Migration** tab in the admin menu.

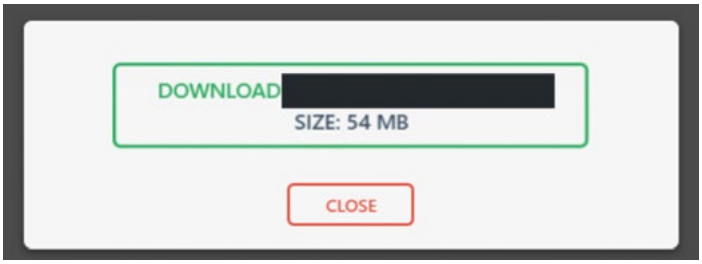
You can run a **Find and Replace** on the database during the export, but you should only do this if you are planning to change the domain of the site or if you plan to use a temporary domain for testing before switching the DNS.



Make sure to select **File** from the dropdown, then run your export.



You can now download the zip folder that contains a copy of your website. The copy of your site should not be considered a backup, as it does not contain the site's files nor does it contain the file that is specific to the **All-in-One WP Migration** plugin.



This completes the actions on your old server.

STEP 2

Complete Steps 5 and 6 from the Manual Migration Process

Just as in the manual migration process, you will need to add your domain to your new host and create a fresh install of WordPress. In this fresh install, you will need to install and activate the **All-in-One WP Migration** plugin.

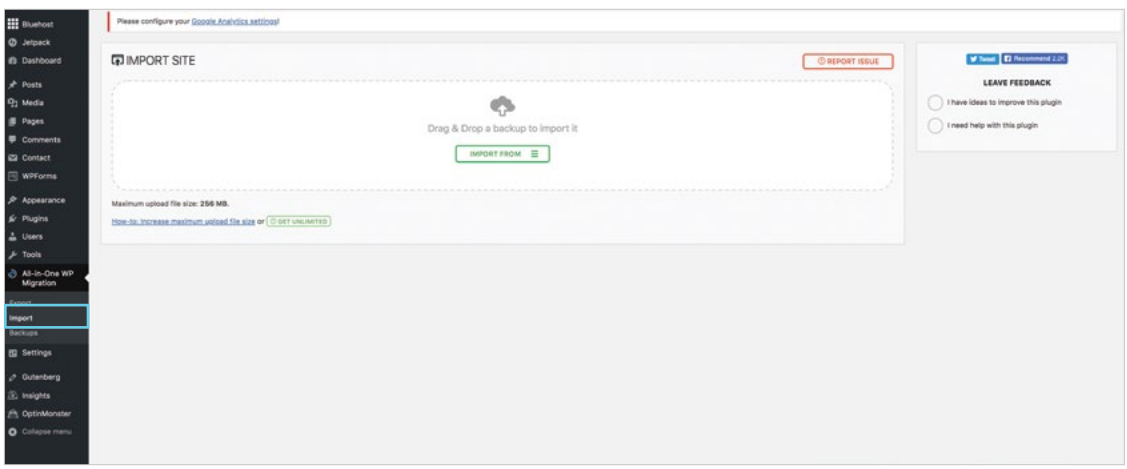
If you plan to use your current domain for your site on the new server instead of a temporary or different domain, you will also need to refer to Step 8 on editing the files from the manual migration process.

If you plan to use a temporary or different domain, then make sure to use the **Find and Replace** feature during the export of your website. This will save you a lot of time and hassle.

STEP 3

Import your website

Go to **All-in-One WP Migration** > **Import**, then simply drag and drop the export of your website and let the magic happen. This may take some time, especially if your website export is large.



STEP 4

Test, Test, Test

Just as in the manual migration, you need to test all the pages on your site and make sure that everything looks as it should and that all the images were imported properly.

STEP 5

Switch your DNS

Follow the instructions from Step 9 of the manual migration.

SECTION TAKEAWAYS

In summary, here is a checklist you can use to improve your migration process.

BEFORE YOU BEGIN

- ○ Get Required Access
  - WordPress Administrator Access
  - Old and New Hosting Account Access
  - FTP Access
- ○ Get the Tools You Will Need
  - FTP Program: Manual Migration
  - Migration Plugin: Plugin Aided Migration

MANUAL MIGRATION

- ○ Log into the current hosting
- ○ Find the database
- ○ Export the database
- ○ Get the files
- ○ Add your domain
- ○ Set up the new WP installation
- ○ Replace the database and files
- ○ Edit your hosts file
- ○ Test, test, test
- ○ Switch your DNS

PLUGIN AIDED MIGRATION

- ○ Install the plugin and export
- ○ Add your domain
- ○ Set up the new WP installation
- ○ Import your website
- ○ Test, test, test
- ○ Switch your DNS

SECTION SIX

ADVANCED

06

ADVANCED DEVELOPER TOOLS

As you fine tune your developer skills, you may want to take advantage of some advanced techniques that can help you manage and maintain your WordPress websites.

Tools like Git, a version control system for tracking files, and WP-CLI, a robust command line tool for interacting with WordPress sites, can save you time and increase your productivity.

Git: Fast, Easy Version Control

Git is a version control system for tracking when files are added, deleted, or changed. Git not only tracks when a file is changed, but who made the changes, and what specific lines were changed. Git also provides tools you can use to revert a project back to any saved point in time.

You may choose to use Git for a variety of reasons. It is incredibly helpful for collaborating with other developers, working on multiple changes simultaneously, handling automated tests for code, deploying code updates, and keeping a granular history and backup of a project at each version.

It's advisable to have any code that is critical to a website — whether it's a plugin, theme, or a command line tool — in a Git code repository.

Git Basics

Like web hosts, there are Git hosts, many who will host your code repositories for free and charge for premium features. Some popular hosts for Git repositories include GitHub, Bitbucket, and GitLab. Each runs the core Git software but adds its own workflow tools for issue tracking and project management.

Notably, Git is intended for managing code files, not your WordPress database. If you need to sync data between environments, we recommend using WP-CLI, a command line interface (CLI) which we'll discuss later in detail, or some other premium solution.

Like writing code, there are many ways to apply Git to your workflow. But before we dive in to common Git workflows, here is a quick overview of some key concepts and vocabulary.

REPOSITORIES

A repository (also known as a repo) is a folder where every WordPress file and folder is tracked by Git. Repositories can hold any kind of file, including code files like CSS and JavaScript, images, zip archives, and software installation files. Some repositories are public for open-source software, like the one WordPress has for the Gutenberg project, the new editor that is being built in phases. Others can be private and require a user to have permission to view them.

BRANCHES

Git repositories have branches off a central hub that deviate away from the hub and then return back to it. Typically, repositories have a master branch that's considered the primary, production-ready branch that's always showing the latest version of a product. They also have a develop branch where the beta work between versions is stored. When you want to work in a Git repository, you create a new branch for each new feature or bug fix, which may often branch off a develop or other feature branch. We'll cover more on this workflow later.

COMMITS

Commits are similar to saving a file. Commits can be as small as fixing a typo in one file or as complex as hundreds of line changes across dozens of files. Commits are made to branches. A Git repository can be rewound to any commit and commits can be compared with one another to see differences.

MERGES AND PULL REQUESTS

When a feature or bug fix is ready, the new commits on a branch are merged into another branch. Often these merges are initiated via a **Pull Request** (frequently abbreviated as a "PR"), which allows a team to approve, reject, or discuss the changes someone is requesting.

PUSH AND PULL

When a repository is cloned to a developer's computer from GitHub or another hosted repository, this creates the concept of an origin repository and a local repository. The concept is similar to how email servers sync an inbox from the email server to email client software like Apple Mail or Microsoft Outlook. Commits are made to a branch on the local copy of the repository and then pushed to that branch on the remote, origin repository. If there are new commits on the origin branch, those can be pulled to the local repository.

When working with Git, developers can use a CLI in a terminal application, a visual graphic user interface (GUI), or a mix of both. Like email clients, there are lots of Git GUIs for Windows, macOS, and Linux that often show commit history in a similar inbox-style fashion to email. These GUIs are a great starting point if you're unfamiliar with CLIs and they can often handle advanced features of Git as well.

Some tasks are faster in a CLI, some are faster in a GUI, and they may vary depending on personal experience. It's okay to have some team members using a GUI and some using the CLIs. What you use and when often comes down to personal preference and comfort.

GitFlow

There are different ways to work with Git. GitFlow is a battle-tested approach to using Git as an individual or as a team to prevent headaches that often arise in software development like rapid bug fixes and security releases. While we recommend trying GitFlow, it's more important that your code is in version control rather than how you work in version control.

In GitFlow, there are two primary branches: master and develop. Master should always be a pristine, production-ready branch that could be used to re-release your website at any time if a server crashes. Develop should ideally be finished and tested code, although if it's not quite ready to release to servers, it could be considered a beta.

New features typically branch off develop, or another feature branch that branched off develop. Changes are committed to those feature branches, tested while on the feature branch, and merged back into the develop branch when complete.

Bug fixes and security patches typically start a new branch off master, which then get merged into both master and develop once complete.

Git in Action with Themes

Let's say you start working on an existing WordPress theme that's stored in a Git repository, and you've been asked to create a new page template. Here are the steps you would take using Git for themes. Note, you can also use the same steps for creating new plugins.

1. Clone the remote repository to the themes directory in a local development environment on your computer.
2. Switch from the master branch to the develop branch.
3. Create a new branch called **new-theme-template**
4. Create the new template file, test the code, and commit the changes on your local repository.
5. Push the local branch to the origin repository.
6. Open a **Pull Request** to merge your branch into develop. This can trigger automated tests to make sure you followed proper coding standards and that the code works with both recent and older versions of WordPress.
7. Ask another developer to review the **Pull Request** by downloading the branch to test, confirming the changes, and providing feedback or approval.
8. Merge the **Pull Request** into develop, which can be automatically pulled into a staging environment for review.
9. Eventually, you will merge the develop branch into master, a new release is created, and that release can be automatically pushed to the production site.

Some of the workflow tasks above, like automated testing and pushing to environments, don't happen out-of-the-box, but they illustrate the power of Git in your workflow. Even the best, most experienced developers can forget to run tests. With practice, almost anyone can manually copy files to staging and production environments.

By investing time to automate these tasks into your workflow, you can have more confidence in your release and collaboration process. You'll be able to catch a missing semicolon or another bug well before they get introduced into your live website. A well-oiled GitFlow leveraging tests and automated deployments can save you time in your day-to-day workflow and also give clients peace of mind that there are safeguards in place to protect their sites.

Git for Updates

Our recommended solution is GitHub Updater, a free open-source WordPress plugin that connects to GitHub, Bitbucket, GitLab, and other Git repository hosts to sync a WordPress plugin or theme with a repository.

GitHub Updater ties into the standard WordPress updates system, presenting updates for custom plugins and themes the same way updates are presented for free products installed from WordPress.org. With two additional lines added to a theme's style.css file or a plugin's main file, you can tell GitHub Updater which repository the product pulls from and which branch to use. Using a personal access token, GitHub Updater can be used to pull private repositories as well as public.

GitHub Updater uses WordPress Cron to check for updates and allows you to manually pull a fresh copy from your Git repository at any time. It also allows you to switch branches, which can be incredibly useful to check a feature branch in a staging environment.

A more traditional option is to install Git as a CLI on your web servers and set a server cron job to pull new tagged releases or commits from a master branch. This can be slightly more reliable, as it doesn't rely on WordPress Cron, which is only triggered if the site front end or dashboards are visited. Git isn't seamlessly integrated into the WordPress admin dashboard and requires you to log in to your server using a terminal to manually run updates or switch branches.

WP-CLI: Fast, Powerful Site Management

To manage your site, you can use WP-CLI as a command line tool instead using the visual WP admin dashboard in a web browser. WP-CLI uses a text-based syntax of commands — combinations of words, numbers, abbreviations, and punctuation — to interact with a WordPress site.

Here's an example of how you would take advantage of WP-CLI using a terminal app or shell:

1. Change directories (**cd**) to the folder where your WordPress site is installed.
2. Type a command using WP-CLI's syntax, such as `wp maintenance-mode activate`
3. Hit enter or return.
4. When the command is done running, WP-CLI will show a written response explaining what operations it ran or errors it encountered.

When you send a command with the WP-CLI syntax, WordPress executes based on the specific commands. There are 44 commands that come with WP-CLI, many with subcommands and flexible options. Each command maps to different features of WordPress like post, option, maintenance-mode, and user.

PRO TIP

Think of WP-CLI as a virtual assistant — like Slackbot or Siri — that you can direct message with to manage your WordPress sites.

Users and Directories

There are two critical concepts to understand while you're using a command line tool like WP-CLI in a terminal window.

First of all, in a terminal session, keep in mind:

- You're always using a user account with varying permission rights to read or write to parts of the filesystem.
- You're always inside a folder or directory somewhere on the filesystem.

When you open a terminal window on a Mac or Linux machine you'll see something like this: `username@hostname:directory[separator]`. As an example, let's use: `bmack@mbp:~$`. Here is an explanation of what you're seeing:

- The system user's name is `bmack`, which differs from the user's display name of Brian Mack.
- The name of the host machine is `mbp`.
- The `~` is a symbolic shorthand for the user's directory (/Users/bmack).
- The `$` is the prompt part of the command prompt that separates a command so you know the terminal is ready for input and has a cursor following it. On Windows, you would see `>`.

We'll get into more details below about how to navigate the filesystem, but for now, know whenever you see that string, it means your terminal is awaiting input and no processes are running.

Navigating Directories and Common Commands

As mentioned above, when you first open a terminal window, you are in your user's home directory.

- The `~` is short for the user's home directory (for example, /Users/bmack).
- The `.` is short for the current directory.
- The `..` is short for the directory above the current directory.
- The `/` at the beginning of a path stands for the root directory of the entire filesystem (elsewhere it denotes the separation between directories). With paths, similar to URLs, there are relative paths and absolute paths.

This is best explained by introducing the `cd` command, which is used to change directories. For example, if you wanted to change a folder called `playground` in the directory of the above user, you would:

- Use `cd playground`, but only if you're in the root of the home directory. This relative path is relative to your current directory.
- Use `cd /Users/bmack/playground` from anywhere (including the home directory). This absolute path can be used anywhere and doesn't take the current directory into account.

As directories change, you'll see your prompt prefix change as well, like so: `bmack@mbp:~/playground$`

Then, if you wanted to step up one directory to return to the user directory, you could run `cd ..` or `cd ../..` if you wanted to step up two directories into the `/Users directory` from `/Users/bmack/playground`

QUICK TIP

For a long directory name that's inside the current directory (such as, **my-super-long-folder-name**), you can type the first few characters `cd my-` and hit the **TAB** key to auto-complete the name without typing it all out. This also works with absolute paths:

- Use `cd /Users/bm` **+ TAB** to get `bmack`
- And use `cd /Users/bmack/my-` **+ TAB** to `get my-super-long-folder-name`

Here are a few common commands and operators:

- **cd**: Change directories
- **ls**: List files
- **touch**: Create a file
- **mkdir**: Create a folder
- **rm**: Remove a file or folder
- **&&**: Chain commands together

Here are some examples of how to use them:

- Change into the themes directory and make a folder called custom: `cd wp-content/themes && mkdir custom`
- Change into the plugins directory and force remove Hello Dolly and recursively remove any subfolders and nested files inside hello: `cd wp-content/plugins && rm -rf hello`
- Change up one level from the current directory and make an empty file called LICENSE.md: `cd .. && touch README.md`
- Change into the "public_html" folder, where WordPress lives, and list files, including hidden system files/folders that begin with a period: `cd ~/public_html && ls -a`

MacOS TIP

For a long directory path, you can type `cd` (note the space) and then drag-and-drop a folder from **Finder** into the terminal window, which will insert the absolute path to that directory.

It can save a ton of time over typing a path like `cd /Users/bmack/Sites/VVV/wordpress/public_html/wp-content/plugins/src/scss`

Remote Connections with SSH

SSH stands for Secure Shell and works like a remote desktop for the Terminal.

SSH lets you connect to a shell session on a remote server with credentials. The server could be elsewhere on the internet, or simply a virtual server running on your local OS like VVV or Lando.

To access WP-CLI you will often need to "ssh in" to your server to begin running commands.

To use SSH you'll need:

1. A **username** for a user on the remote server
2. A **hostname** for pointing your computer at the server (either an IP address or domain like something.mydomain.com)
3. A **password** or **key file** used to authenticate as that user.

You may also need a port number, if the default SSH port 22 is blocked.

Using the `bmack` example, here's what the above information may look like:

```
bmack@mbp:~$ ssh brian@192.168.32.145
password: **********

brian@192.168.32.145:~$
```

If you've set a default user and key file in your SSH config, you can use this:

```
bmack@mbp:~$ ssh 192.168.32.145

brian@192.168.32.145:~$
```

A few things to keep in-mind when you're SSH'd into a remote server:

- You don't have access to command history from your local system.
- You don't have access to any aliases in your local system's `~/.bash_profile` — You have your remote user's `~/.bash_profile`
- You only have access to software installed on the server, not on your local machine. For instance, you may be able to run `npm install` on your local machine because you installed **npm**, a JavaScript development tool, on there. But that command may not work on your remote server unless **npm** is installed there.

Getting Started With WP-CLI

WP-CLI is likely already installed on your web host, as it's typically pre-installed on hosting servers. WP-CLI is also bundled with popular local development tools like VVV, DesktopServer, and Lando.

The first time you use WP-CLI on a Bluehost account, you'll need to call support and request shell access. Once you've received access, you can use it on any site on your account — including future sites you create.

To start using WP-CLI, you'll likely need to connect to your web server or virtual server using SSH.

1. Log into your server using SSH.
2. Use `cd` to get into the WordPress root directory, where you'll find things like `/wp-admin` and `/wp-content`
3. Type `wp` and press return on your keyboard.
4. If WP-CLI is installed correctly, you should see the help guide with a list of available commands on the server.
5. Type `q` to quit the help documentation.

Commands and Subcommands

At the center of WP-CLI are the commands. At the time of writing, there are 44 top-level commands included with WP-CLI to run operations on data in a WordPress database, generate WordPress starter code, and more!

Commands are pretty easy to identify — they're the first word following `wp` in a command string, such as `wp post`

When commands are registered with WP-CLI, a keyword string is mapped to a callback to execute when that command is used. Here is an example:

Type `wp post` to work with Posts, type `wp option` to work with WordPress Options, and type `wp help` to learn about available commands.

Subcommands are registered beneath a primary command, such as `wp post delete`. They are commands that aren't top level (top-level commands are immediately right-adjacent to the `wp` command), but they have all the same potential features as a top-level command.

It's not a hard rule, but in general, the primary command is a thing (such as a post, user, cron, or config), while subcommands are operations that run on that thing (such as create, delete, start, or install).

It's also important to note that while `wp post create` and `wp user create` both have a `create` subcommand, these are two distinct subcommands with different arguments and functionality.

Arguments and Flags

Many WP-CLI commands have required or optional arguments (args) that change how the commands work, scope what the operation will modify, and return a response. Flags are used to enable or disable command features.

POSITIONAL ARGUMENTS

Positional arguments are values that immediately follow the command or subcommand keyword, separated by a space. The order of positional arguments matters, hence the name positional.

For example, in `wp post delete 100`, 100 is the first positional argument following the delete subcommand. In a simple top-level command like `wp do-something my-plugin-name`, `my-plugin-name` is the first positional argument, not a subcommand. Here's an example of multiple positional arguments on a subcommand:

```
wp do-something-else subcommand arg1 arg2 arg3
```

FLAGS

Flags are similar to checkboxes in a visual interface — they're used to denote true/false and are used to enable or disable part of a command's functionality. They're called flags because their syntax looks like a flag pole, in that the word is "flying from" with two preceding dashes. For example, in `wp post delete 100 --force`, `--force` is the flag.

ASSOCIATIVE ARGUMENTS

Associative arguments are values that are associated with a string key. These key-value pairs start with a flag syntax, but an `=` is used to set a value. Unlike positional args, associative args are always optional and can be used in any order. Associative is often abbreviated association in code. Here's an example: `wp post delete 100 –method=via-sql-query`

GLOBAL ARGUMENTS

At the time of writing, there are 13 global arguments that can be used with any command included in WP-CLI or a custom command. All global arguments are associative args, as they're all optional.

Two critical global arguments to be aware of that greatly expedite the learning and use of WP-CLI are:

1. `--prompt`: This creates an interactive prompt that asks a user for values for each positional argument, associative argument, and flag. This means you won't have to memorize argument keys or the order of arguments. Hitting return with an empty value in `--prompt` will fall back to the default value for the argument, as if you didn't type it.

2. `--help`: This will return the documentation for any command or subcommand. For instance, `wp post --help, wp post list --help`, and `wp maintenance-mode --help` will all return different help responses. You don't need to go search the web for information about commands!

Common Uses for WP-CLI

Here are some common things you can do with WP-CLI:

- Import or export your entire WordPress database (`wp db`).
- List, import, or export WordPress data such as posts, users, comments, and options in a variety of data formats like JSON, CSV, and YAML.
- Run a **search and replace** operation on your WordPress database for converting URLs between environments, updating option, or meta keys (`wp search-replace`).
- Set up user roles and capabilities (`wp user` and `wp cap`).
- Start new plugins or themes (`wp scaffold`).
- Flush rewrite rules, WordPress transients, or object cache.
- Execute MySQL queries or PHP from the command line to accomplish a one-time task.

Chaining Commands Together

Commands don't run in parallel, so they can be chained together and expected to run sequentially, as seen in this example:

```
wp cli update && wp core update && wp theme update
--all && wp plugin update --all
```

One risk of running commands in a sequence like this is that if one command errors-out, subsequent commands are never run. So, if you set a cron job to run updates daily, but each day the WP-CLI update command fails, none of the other commands after it will run. So when chaining commands together that aren't related like above, consider the order of importance and put the most stable or reliable commands first.

Composing Commands Together

Some of the real power of WP-CLI is composing commands together in a sequence where subsequent commands use the dynamic responses from preceding commands.

By making commands that focus on doing one thing, the results of one command can be passed into arguments for another command, making them much more versatile and composable.

There are two primary ways to compose commands together:

1. Using a subshell to use the results of a command inside another command.
2. Use `xargs` to pass response items to the second command.

In general, `xargs` syntax is easier to read, but there isn't a right way to do it, and sometimes there are benefits to subshells and loops.

Command Example: Deleting Drafts

The WordPress admin dashboard is a fairly powerful interface, but it does have limitations. Using the filters on the **All Posts** screen, it's possible to get a list of all draft posts or posts by a specific author, but not a list of all drafts by a specific author.

On a news website, a user might have thousands of posts and hundreds of drafts, which would make cleaning drafts out for that user an exhaustively manual task in the WordPress admin.

Here are two quicker options with WP-CLI:

USE A BASH VARIABLE

```
wp post delete --force $(wp post list --post_
status=draft --post_author=1 --format=ids)
```

USE XARGS

```
wp post list --post_status=draft --post_author=1
--field=ID | xargs -n1 -I % wp post delete % --force
```

On a news website, a user might have thousands of posts and hundreds of drafts, which would make cleaning drafts out for that user an exhaustively manual task in the WordPress admin.

> You should now have a foundation of understanding when it comes to troubleshooting issues, streamlining workflows, increasing security measures, and more. Use this guide to implement best practices, tighten up your workflows, and become a more thorough web pro.

SITE LAUNCH CHECKLIST

Use this checklist as a step-by-step guide to launching a new website as a web pro.

◯ PLACEHOLDER CONTENT

Many sites will use filler text or placeholder images during the site set up and testing phase. Theme content demos can also add generic content with which you would not want to launch. Look for and remove or update any placeholder content.

◯ PROOFREAD

Bad grammar and spelling mistakes can detract from the professionalism of your site. Be sure to get a few people to review the content for mistakes.

◯ BROKEN LINKS AND IMAGES

Use a tool like Screaming Frog to scan your entire website for invalid links and missing images.

◯ CONTACT PAGE

Don't forget to create a contact page so that visitors know where your business is located and how to contact you. The WP Forms plugin makes it easy to get up and running with a contact form.

◯ DOWNLOADABLE FILES

Make sure to test out any file downloads by verifying that links are pointing to the correct file and that the files are in fact downloading.

◯ ANALYTICS

Data about who is visiting your site, what content is most popular, and what keywords are generating the most traffic will help you make educated decisions regarding future content and marketing. Ideally, integrating Google Tag Manager on your site makes it easy to inject the Google Analytics tracking code or other code snippets onto your site.

DESIGN

◯ LOGO

Your logo should be prominently displayed in the site's header and should link to the homepage.

◯ FAVICON

A favicon is an icon that shows on a browser tab right next to the page title. This can help your site's branding and professionalism. Just open up the theme customizer in WordPress, navigate to the Site Identity section, and then upload an icon to the Site Icon section.

◯ BROWSER COMPATIBILITY

Preview your site in the most popular browsers such as Chrome, Safari, Firefox, and Internet Explorer/Microsoft Edge. Look for layout issues or an inability to see or interact with certain elements on the page such as video and audio. Use tools like Browsershots or BrowserStack to simplify this process.

◯ RESPONSIVE DESIGN

Test your site on mobile phones and tablets to make sure the site is easy to use and links are big enough for fat fingers to tap on. Don't have a specific device? Try out Google Chrome's Device Mode, which lets you emulate a number of popular phones and tablets.

◯ VALIDATE MARKUP AND STYLES

Invalid HTML markup or CSS rules can cause browser rendering issues or prevent search engines from properly processing data on web pages. Use the W3C Markup Validation Service to check HTML and the CSS Validation Service to check for CSS incompatibilities.

◯ PRINT STYLESHEET

Use the print preview feature in your browser to see how pages will look when printed. If adjustments are necessary, use a print stylesheet to customize the view.

◯ ACCESSIBILITY

Much like providing wheelchair access to a public building, your website needs to be user-friendly for those who may have various handicaps. Download the Axe Accessibility Checker for Chrome to check for issues on your website.

◯ CONTENT OVERFLOW

Try adding really long titles and descriptions to pages or menu items to see how your theme handles them. In some cases, long URLs or code snippets can overflow and interfere with other elements on the page.

FUNCTIONALITY

◯ TIMEZONE

Be sure to visit **Settings** > **General** in the WordPress admin and set the timezone to avoid issues with calendar and booking plugins or scheduled posts.

◯ 404 PAGE

Visit a URL on your site that doesn't exist and make sure your 404 (Page Not Found) page matches your theme and is helpful for users who may land on your site from a faulty link.

◯ SITE SEARCH

Perform a search on your website to make sure your site's search results page looks good and returns the expected results.

◯ FORMS

Be sure to test any web forms on your site to make sure form confirmation messages are helpful, data is correctly stored, form emails are sent to the right people, and thank you pages are properly displayed.

◯ EMAIL

Some hosts will completely block the sending of emails directly from the server for security reasons. In order for emails to be properly delivered, be sure to set up all outgoing emails to use SMTP. If you have a business, look into transactional email services.

◯ SOCIAL INTEGRATIONS

Check to see if social integrations are configured and working correctly. This includes making sure the correct profiles and platforms are in place. Also, check meta tags for social to ensure that shared pages appear correctly on each social platform.

◯ THIRD-PARTY INTEGRATIONS

It is common to integrate a site with third-party services like a CRM, merchant gateway, or marketing automation service. Test each service to make sure they work as expected. Don't forget to do a test purchase on eCommerce sites and make sure you didn't forget to turn off test mode when you launch.

◯ SITE HEALTH CHECK

In the WordPress administrative menu, go to **Tools** > **Site Health** to visit the Site Health page. This is where WordPress, plugins, and themes can automatically detect and surface known issues related to the configuration of your website. Be sure to address any critical issues.

◯ SPAM PREVENTION

Any form on your site can be a source of spam. Spam comments and contact form submissions are the most common. Be sure to implement some form of CAPTCHA for important forms and use a plugin like Akismet or Antispam Bee to prevent spam comments.

◯ DISABLE "COMING SOON"

If your site has been showing visitors a **Coming Soon** page during development, be sure to disable this so that everyone can see the site.

◯ CONFIGURE UPTIME MONITORING

Set up a monitoring tool like UptimeRobot to send a notification so you are the first to know in the event that the website goes down.

SEO (SEARCH ENGINE OPTIMIZATION)

◯ PLUGIN

WordPress handles SEO fairly well out-of-the-box, but it makes many assumptions that may not always be correct. Installing a SEO plugin will give you control over some very important settings that you would otherwise not be able to manage. The Yoast SEO and All-in-One SEO Pack are both excellent options.

◯ PERMALINKS

Visit the **Settings** > **Permalinks** page in WordPress to configure the preferred URL structure for your site. Make sure that each page URL contains the primary keyword or phrase.

◯ REDIRECTS

If you are migrating a site to a new domain or changing the URLs where pages are found, redirects help ensure that search engines can find the new page and that the SEO value of the old page is carried over as well. Some excellent plugins include Redirection, Safe Redirect Manager, and Simple Website Redirect.

◯ SITEMAP

A sitemap helps search engines find all of your content and can also help manage how frequently certain pages are crawled. Most SEO plugins provide the ability to configure a sitemap.

◯ SITE TITLE AND TAGLINE

Make sure your site's title and tagline have been properly set under **Settings** > **General**.

◯ PAGE TITLES

Every page or post should have a unique title that is less than 70 characters and contains the primary keyword or phrase.

◯ META DESCRIPTIONS

While not required, meta descriptions are used by search engines and are typically what is shown below the title in the search results. If not set, search engines will select a portion of text on the page that best matches the current query. Well crafted meta descriptions may be used in the place of dynamically generated descriptions and can help attract visitors to click through to your site. Meta descriptions should be no longer than 155 characters.

◯ OPTIMIZE IMAGES

Search engines can't look at an image and tell what it is about, so they use context instead. Make sure that the file names, descriptions, and ALT tags all contain the appropriate keyword or phrase and that any text located near the image reinforces the keywords and subject of the image.

◯ STRUCTURED DATA

Using structured data on your site helps search engines to identify and properly index specific types of information. Everything from people and places to addresses and recipes can leverage structured data. Consult schema.org for more details and use Google's Structured Data Testing Tool to validate your implementation.

◯ WEBMASTER TOOLS

Connect your site to Google's and Bing's webmaster tools so you will get notified in the event that one of these search engines detects a problem with your site.

◯ ALLOW SEARCH ENGINES

Look under **Settings** > **Reading** in the WordPress dashboard and make sure to uncheck the box labeled **Discourage search engines from indexing this site**.

◯ REVIEW ROBOTS.TXT

If you've created a custom `robots.txt` file, be sure to delete or edit it so that search engines aren't being told not to crawl and index the site.

SECURITY

◯ SOFTWARE UPDATES

Make sure you are running the latest version of WordPress, as well as your plugins and themes. If any software requires a license, be sure to enter the license key so you won't miss out on future updates. Running old versions often makes you a target for hackers.

◯ BACKUPS

In the event of data loss, a hacked site, or other catastrophes; having a safe, clean backup is imperative. Backups should be automatic, regularly scheduled, redundant, and stored off-site. Be sure your backup schedule is more frequent than your editing schedule and pay close attention to your backup retention policy. Verify your backups to make sure they are complete and stored in the correct location. Use a plugin like Updraft Plus to backup your site to services like Dropbox, Google Drive, or Amazon S3.

◯ USE SECURE PASSWORDS

Ensure that every user that can log in to the site has a secure password; the longer the better. Depending on the processing power available to an attacker, an 8 character password can be cracked within 5 days to 7 minutes. Simply using a 10 character password can increase this time to 3 years.

◯ LOGIN PROTECTION

Hackers often use software to try and guess your password, which can be very effective if you don't have a way to slow down their automated software. One way of doing this is to block a particular IP address after 3-5 failed login attempts. Use a plugin like Login LockDown to set this up.

◯ DISABLE THE THEME AND PLUGIN EDITOR

Make sure that your site's `wp-config.php` file contains the following code:

```
define( 'DISALLOW_FILE_EDIT', true );
```

This will prevent anyone who has gained access to your site from being able to edit files and inject malicious code.

◯ ENABLE SSL OR TLS

If your website doesn't have https at the beginning of the URL, then you do not have an SSL (secure socket layer) or TLS (transport layer security) certificate set up. Without this, traffic to and from your site could be accessed by others. This is mandatory for any site where credit card sales take place, but can also help boost your site's SEO.

◯ DELETE INACTIVE PLUGINS

Plugins that aren't actively being used can still be a security issue, even though WordPress doesn't run the code. Since you aren't using them anyway, just delete all inactive plugins.

◯ REMOVE NON-MANDATORY PLUGINS

Don't forget to remove plugins that are one-time use only or that are used to debug or prepare a site for launch. If there are any plugins that you can live without, consider removing these as well.

◯ CHECK FOR VULNERABILITIES

Use the WordPress Vulnerability Database to check your theme and plugins for known security vulnerabilities.

◯ DISABLE DIRECTORY BROWSING

Directory browsing allows anyone on the web to access the contents of a folder on your server. This presents a security risk and may allow people to find and download private files. Disable this feature in Apache by adding `Options All -Indexes` to your `.htaccess` file.

PERFORMANCE

◯ COMPRESS IMAGES

Don't upload enormous images. Make sure to size them down to something reasonable before uploading them. Use a plugin like Smush to ensure that all existing images are compressed and that all images uploaded in the future are also optimized for the web.

◯ ENABLE OBJECT CACHING

If your web host supports object caching, be sure that it is enabled. This can significantly reduce the number of queries that WordPress needs to run and can speed up page loads.

◯ ENABLE RESOURCE CACHING

Caching entire web pages and/or important files can significantly boost not just the speed of a site, but also its ability to handle concurrent requests. Managed web hosts tend to handle resource caching for you, but there are some excellent plugins such as WP Rocket, WP Super Cache, and W3 Total Cache.

◯ SET UP A CDN

A CDN (content delivery network) will store a copy of certain files or your entire site in various locations so that when visitors make a request the content can be served as fast as possible. Managed web hosts typically provide a CDN. JetPack also offers Site Accelerator which is a free CDN for images and static files.

◯ SITE SPEED AUDIT

Use Google's Page Speed Insights to get an idea of what performance issues are affecting your site.

◯ ADD A DATABASE INDEX

Every time WordPress loads a page, it performs a special query to fetch all of the options in the database that should be autoloaded. However, the WordPress database doesn't have an index on the autoload column in the options table to speed up such queries. Adding an additional index to this column can significantly improve the speed of this query and shave time off of page loads.

◯ CHECK DATABASE STORAGE METHOD

New WordPress installs should already be set up to use the InnoDB MySQL database engine for all of the tables in the database. However, older sites may still be using the MyISAM storage engine or a combination of both. For optimal results, ensure that all tables in the database use the InnoDB storage engine.

LEGAL

◯ ACQUIRE LICENSES

Ensure that all software or resources used on the site can be freely used or have the appropriate licenses. Be sure to review the licensing for plugins, fonts, and images.

◯ COPYRIGHT NOTICE

Provide a copyright notice in the site's footer.

◯ PRIVACY POLICY

A privacy policy tells people how you plan to use the information you collect from them and what types of information will be collected. Typically, a privacy policy will live on its own page and is linked to in the footer menu.

◯ TERMS AND CONDITIONS

If you are running an eCommerce site or deal with money in any way, be sure to clearly outline the terms and conditions on your site. This should include things like return policies, guarantees, payment terms, and delivery timelines.

◯ COOKIE CONSENT

The GDPR (General Data Protection Regulation) passed by the EU applies to any site that collects personal information and may have visitors from the EU. Since storing a cookie on a visitor's machine is an act of data collection, the law dictates that the user's consent is required. As a result, most sites have integrated a cookie consent banner in order to comply.

GLOSSARY

.htaccess

.htaccess file is a granular configuration file for the Apache web server software, used to set or alter the server's configuration settings for the directory in which it is present, or its child directories.

A

Application programming interface (API)

APIs are sets of subroutine definitions, communication protocols, and tools for building software.

B

Backups

Backups help protect your website in the event of server meltdown or errors.

C

Caching

This is a process that saves data temporarily so a website or browser doesn't have to download it each time.

Cascading style sheets (CSS)

CSS is a web language that allows stylistic elements to be applied to HTML without editing them into the HTML itself.

Clone production

This process can re-copy all your files and data from your production website to staging with new updates.

Content delivery network (CDN)

CDN is a network of servers that deliver cached static content from websites to users based on the geographic location of the user.

Content management system (CMS)

A CMS is an integrated set of tools and strategies for developing and delivering content over the internet.

D

Database

This is software that is used to manage information in an organized fashion. In WordPress, databases can help store and retrieve blogs, posts, and comments.

Developer

A developer, or dev, is a computer programmer who is active in creating, modifying, and updating a software project.

Domain name servers (DNS)

DNS is a system that maps domain names to IP addresses.

F

Firewall

A firewall is a security system that acts as a barrier between trusted and untrusted traffic.

File transfer protocol (FTP)

FTP is an internet protocol used for transferring files. It can be used to download files or upload files to a server.

G

Git

This tool is a version control system that can be used locally, privately, or with online services. Git is used to manage changes in code.

Graphical user interface (GUI)

GUIs are interfaces that allow users to point the mouse or cursor to graphical icons.

H

Hypertext markup language (HTML)

HTML is the standard markup language for creating web pages and web applications.

J

JavaScript

This is a programming language that WordPress uses to make certain processing occur in a web browser when it is convenient or impossible for the server to do that processing.

L

Local environment

Running WordPress in a local environment means the software is running on your computer instead of relying on a server on the internet.

M

Metaboxes

Metabox creation is essential for WordPress theme and plugin development. It helps users add an editor to the post screen instead of relying on custom fields.

P

Preprocessor (PHP) programming language

PHP is an open source programming and scripting language used to create interactive websites on WordPress.

Plugin

A plugin is a combination of PHP, JS, CSS, and other assets. Plugins can be added to a WordPress website to give additional desirable functionality.

Post type

Post type refers to the various structured data that is maintained in the WordPress posts table.

S

Secure Shell (SSH)

SSH is a communication protocol for connecting to remote computers over TCP/IP.

Secure Sockets Layer (SSL)

SSL is the predecessor to Transport Layer Security and provides cryptographic protocols for secure communications across an unsecured network like the internet.

Shortcode

Shortcodes can embed files or create objects in a single line that would normally require a lot of complicated code.

Staging environment

A staging environment contains all of the files in WordPress as well as plugins, themes, and uploaded media files. It also includes a full copy of the WordPress database containing posts, users, site options and other data.

Structured query language (SQL)

SQL is a programming language used to manage databases. MySQL queries are used in WordPress to get data and generate web pages.

T

Template

This is a file that defines an area of the web pages generated by a theme.

U

URL

A URL is an address of a specific web site or file on the internet.

W

Web application firewalls (WAFs)

These firewalls provide a level of protection against common security issues.

Website migration

A website migration can occur when there is a change to the site's location, platform, structure, content, design, or user experience design.

Web server

A web server provides leased hosting that houses and servers websites. It can serve HTML but can also serve HTML generated by WordPress or other dynamic languages.

X

xargs

xargs is a command on Unix and most Unix-like operating systems used to build and execute commands from standard input.

WANT TO BUILD YOUR OWN WEBSITE?

WordPress provides the ideal ecosystem for any web pro looking to expand their personal or online goals. *The Blueprint: A Web Pro's Guide to WordPress*, is the perfect place to begin that journey.

This guide can help you kick start your publishing efforts with a few simple, straightforward instructions. Step-by-step, *The Blueprint* provides an overview of the most common pro features of WordPress, along with instructions on where to get started.

Get started today and you'll be a webmaster faster than you think.

bluehost

Endurance International Group
10 Corporate Drive, Suite 300
Burlington, MA 01803

WWW.BLUEHOST.COM