WP engine®

The ultimate guide to Agile development on WP Engine.

By Janna Hilferty

WHITE PAPER

Table of Contents.

Introduction.

Agile is a term used for a set of principles surrounding iterative, incremental software development methods. It is quickly becoming the most popular way for teams of developers to coordinate and rapidly deliver quality software products. Agile methods allow teams to communicate openly, continually commit changes, and deliver tested and approved code. Open lines of communication increases cross-team coordination, and allows for proper expectation management with your stakeholders. Perhaps most importantly, it helps avoid technical debt and bootstrapped solutions that barely work. In this guide we'll explore the methodology behind Agile development, Agile workflows, and how WP Engine fits into the equation.

Principles of Agile development.

Rapid release for the best end product

The first principle of Agile development states that committing early and often will offer greater customer satisfaction. Agile methods encourage developers to ship code changes "often"--and this can mean several times a day, or several times a week.

Second, the Agile Manifesto states that the end user deserves the best product, and the best, most competitive products often have dynamic requirements. This means developers should view changing product requirements as an opportunity to delight their user base.

Working iterations

The Agile Manifesto goes on to say that iterations of development should all be working models. Some compare this to a "walk, bike, drive" product lifecycle. Each stage may look completely different from the last. But, each will meet the original purpose and needs of the end user, and each iteration does so more efficiently than the last. This also means your team of developers must not only commit often, they must deliver working products often as well. A working product is the true measure of success, and as such, your Sprints should be centered around this goal.

Cross-team collaboration

"Business people and developers must work together daily throughout the project." This is where the cross-team coordination comes into play. Daily standup reports help keep leadership and stakeholders aligned and accountable to each other.

Agile methods also promote driven but sustainable work, which means the work done in any given sprint should be a manageable workload that the team could manage any week of the year. And teams should be centered around driven individuals, built with the support and resources needed to get the job done.

Learn from retrospectives

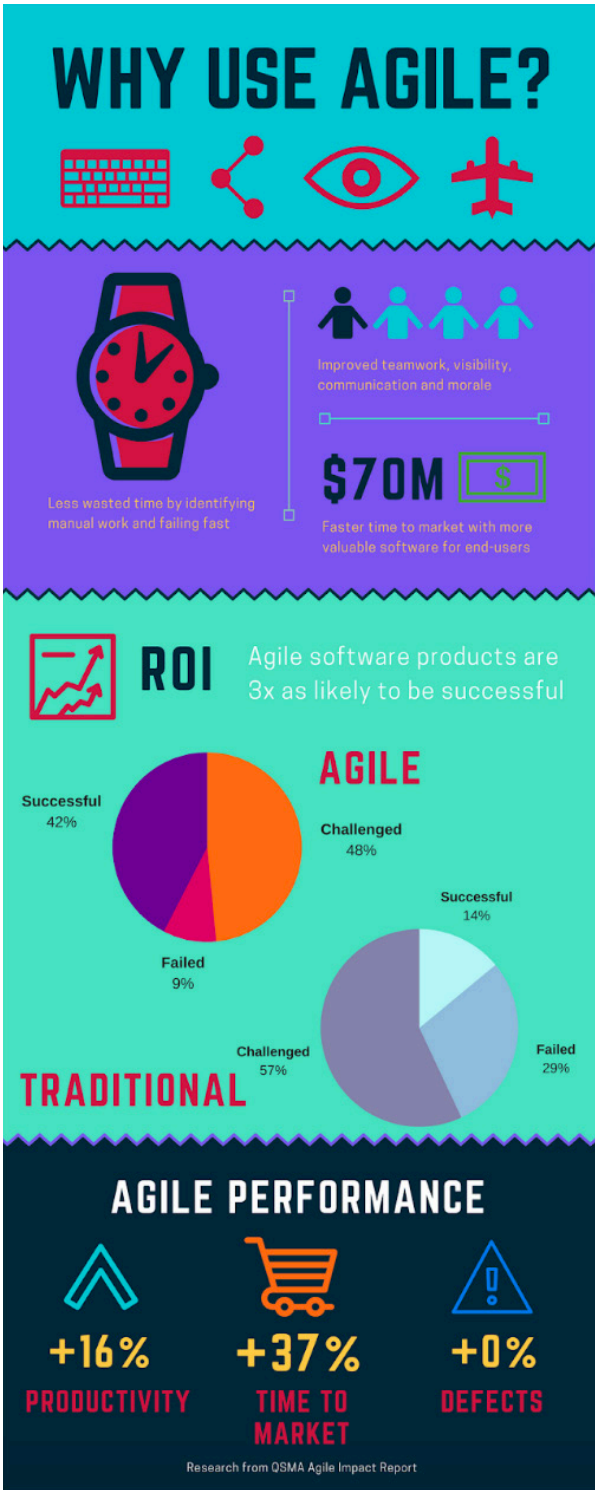
Last, Agile developers meet to reflect on what went right and wrong along the way in Retrospectives. It is important to hash out failures, to ensure the team doesn't repeat mistakes of the past. It also gives the team a chance to examine what went well, so these successes can be integrated into future workflows.

Why use Agile?

The principles alone may be enough to convince some to convert to an Agile workflow, but for those who are not quite as easily convinced, let's review the most important reasons why teams choose Agile.

Faster time to market

The biggest and most obvious win Agile provides is a faster development strategy. With rapid releases and goal posts for minimum viable products (i.e. working products), your time to market is reduced from years down to weeks or months. That means Agile methods can often have a material impact on your business as a whole.



Quality increase

With automated tests, code reviews, and increased communication, your quality of product is certain to increase when using Agile methods. Agile allows your team to hold all code to a set of quality standards before being committed to the repository, and that equates to fewer mistakes, and fewer bootstrapped solutions that break at every turn. It also means eliminating most (if not all) technical debt. Making up for development mistakes of the past can be one of the most costly tasks to take on. When your team uses test-driven Agile development methods from the start, products are developed right, the first time.

Reduce wasted time

One exercise many proponents of Agile encourage for anyone curious about its benefits is to simply sit down, and talk through what you do, auditing your calendars in the process. Many times just talking through redundant tasks your team takes on daily is enough to bring to light where time is being wasted. With Agile, your team can set clear priorities, and easily identify the time which could be saved with minor changes. Not only this, with Agile users also spend less time developing features or changes that never see the light of day. That means more time to develop prioritized, quality features that make a difference for your end-user.

Improved morale

By evening the workload into manageable weekly tasks, and increasing cross-team communication, morale of overworked developers will increase greatly. Gone are the days of spending weeks developing a feature that ends up being scrapped the next week in a review phase. Developers are given the community and support they need to succeed, and are able to manage their workload day to day with ease while still releasing viable, working products regularly.

Delight end-users

A rapid release schedule for quality products is in the best interest of your end user. As the principles state, Agile methods believe your customer deserves the very best product. The flexibility of Agile development means teams can adapt to a market demand or need at a moment's notice. And the speed and agility coupled with high quality products will delight users and increase adoption rates.

Scrum.

Scrum is one of the most common subsets of Agile development used by those creating software products or websites. We will look specifically at Scrum as an example of Agile implementation in practical use. Scrum projects are made up of Roles (Scrum Master, Stakeholders, Team, Product Owner, and Users), Artifacts (Backlogs of requests), and Time Boxes (time allotted to a sprint), which we'll explore further.

Scrum artifacts

The first part to understand about Scrum is how pieces of work are separated and organized. Tasks are categorized into either a ***Product Backlog***, or ***Sprint Backlog***. The Sprint Backlog is for ***Stories***, or groups of tasks which the team reasonably believes can be handled in the upcoming Sprint (generally 1-2 weeks).

A "Story" basically says what the team is building, and why. Some teams prefer to construct each Story by saying, "As a ____ user, I need to do _____, because _____." Each Sprint, multiple Stories can be handled by the team. If a particular Story is too large to be handled in a single Sprint, it is called an ***Epic*** instead, to better represent the body of work to be completed.

The Product Backlog is for user or stakeholder requests or Stories which are prioritized, but not to be handled in the upcoming Sprint. It handles expectations for what should come next in the development process, and as such, it may be an ongoing log that is never completely emptied. The Sprint Backlog is divided up among teams to complete in the upcoming Sprint. And as tasks within the Backlog are completed, they must undergo user acceptance and build tests to ensure quality. Only once these tasks have been completed, tested, and fully deployed can a task be marked as ***Done***.

Scrum roles

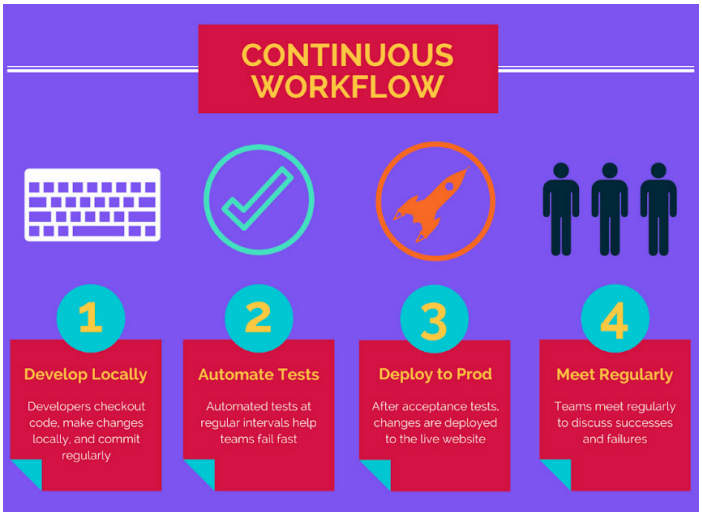
Within a Scrum team, there are several roles which members must fulfill. Agile and Scrum both encourage self-forming teams, which means your team may be in a constant state of flux. Within each team there should be a ***ScrumMaster***, who holds daily 15-minute standup Scrum meetings. The ScrumMaster is in charge of organization and communication between the team and Stakeholders. They also are tasked with removing obstacles and increasing productivity and creativity among their team members. Furthermore, they help the ***Product Owner*** to better

understand how to get the best return on investment from their team by using Agile methodology.

The Product Owner is essentially the one person who owns the "list of requirements" for the product. They also are in charge of prioritizing and setting the order in which new features are released, as well as the schedule for release to customers. This is known as the "User Story," or list of user-facing requirements and timeline for release.

Last, the ***Development Team*** is the self-formed group of developers who do the nitty gritty work of accomplishing the tasks and tests involved in each Sprint. The team divides up the work to be done amongst themselves, based on the amount of work (measured in Story Points) needed to accomplish those tasks.

Agile team workflows.



There are several Agile workflows and systems development teams may use, but all these methods come down to one overarching workflow when it comes to developing for websites:

1. There is a single repository or "source of truth" which you can think of as a library of code.
2. Developers "check out" specific pieces of code (books) to edit, then commit their code back into the repository when complete.
3. When code is committed to the repository, automated tests are run to ensure the code adheres to user acceptance and usability standards.
4. Often, this also triggers a code deploy to update the staging or production website with the changes. If this isn't triggered automatically with each commit, there may be pre-determined intervals for deploys to be triggered (weekly or bi-weekly, most often).

This type of development pipeline is often referred to as "Continuous Integration," in which developers continue to commit changes often and regularly back to the repository. This subjects their code to regular testing and acceptance to ensure they are on the right path. And if the pipeline also includes automated deployments, it might also be referred to as "Continuous Delivery/Deployment," where not only are changes committed and tested regularly, they are also built and deployed at a regular pace. Continuous workflows are imperative to the Agile development process, because they meet the stringent requirements of committing code often and delivering working iterations of products regularly.

Beyond the coding aspect of Agile methods, developers should also adhere to the Agile project management standards. That means in-person "standup" reports daily, cross-team meetings weekly, weekly stakeholder updates, and retrospectives after a project concludes. The project management aspect also includes team organization, which often happens in KanBan or Scrum boards. Tasks are given "story points" which estimate the amount of time and work each task will take to complete. And these points are used in determining what tasks are reasonable to expect for completion in measured weekly or bi-weekly sprints. Remember, Agile development is focused on sustainability. That means the work you take on this week should be a manageable level that you could take on for any week of the year.

The Agile toolbelt.

All of the organization and development behind Agile methodology requires an advanced set of tools. In this section we'll explore the types of tools your team will need to get started.

Task management

The first tool an Agile team needs to succeed is a task management system. How do you track the tasks that need to be completed, measure their time investment, and organize them into teams and Sprints? There are many tools on the market to help your team accomplish the management of these tasks, but we have singled out the best and most-used tools for you below.

JIRA

Jira by Atlassian is one of the primary tools we use in R&D at WP Engine, and it serves these functions well. Atlassian brands Jira as the "#1 software management tool used by agile teams." If your team is relatively small (10 employees or under), the cost is certainly an advantage at only $100 per year. You get a rich set of features for development, including project status, Kanban boards, Story points, and integration with Github, Zendesk, Slack, and many more other tools that increase team productivity. While the feature set is rich for Jira, it may be too intense for non-development teams as it does come with a learning curve.

TRELLO

Trello, by contrast, is all about simplicity. Trello's plans are separated by feature-set and by user, allowing for more powerful tiers of tools for Enterprise accounts while maintaining a more simple set of tools for smaller teams just getting started with Agile. What users tend to love the most about Trello is the ease of use - you start with a blank board and build out from there, allowing users to be as complex or streamlined as they need to be.

Compared to other options on the market, Trello has a relatively low buy-in rate for teams to experiment at the level they choose. And its intuitive, drag-and-drop workflow is user-friendly for individuals and teams alike. Trello is best used by smaller teams who don't have a lot of subtasks within their projects.

ASANA

Last, Asana offers its complete feature set free for teams of 15 developers or fewer. Beyond that, its pricing model is similar to Trello, with unlimited users at a set price-per-user. Asana tends to be best for middle-sized teams who work on small numbers of total projects, which are made up of at least ten tasks. It allows users to assign tasks to themselves or others, set scheduled dates for completion, and manage projects on a dashboard. However, it doesn't allow users to segment projects further, which means dashboards can easily become overrun with too many concurrent projects.

Environment management

Within your development workflow as a team, there's several project stages and environments needed to ensure the success of your Continuous Integration, and automated deployments. Before we dive into the tools, first let's cover some basics of development best practice.

Broadly speaking, an experienced team of developers will never test changes in a production environment--that would risk a negative experience and potential downtime for your end users. That means at a minimum, there should be a sandboxed development environment to test code, and a staging environment to ensure your code behaves the way it should before copying the changes to production.

At WP Engine we make the management of these stages easy. Each Site is broken out into three Environments: Development, Staging, and Production. While your live website lives in the "Production" environment, you can have your working "Development" version as well. From the Development environment users can "check out" code, and merge their changes back in using our git system. And the "Copy To/From" tool will help you easily publish those changes to Staging, and then Production when the changes have been approved.

On top of these environments, your team can add automated Continuous Integration and deployment tools to add further building and testing tools to fully flush out your Agile utility belt.

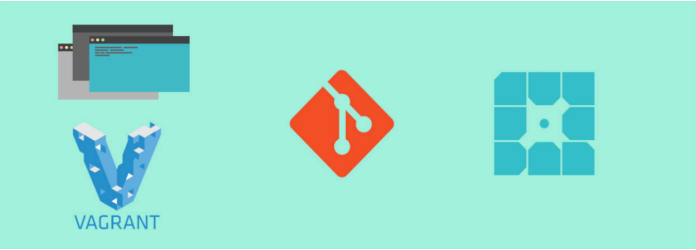
Version control

We have already covered the need for a "single source of truth" when it comes to your team's codebase. When developing with a team, version control over this codebase is paramount to your team's success. Version control simply allows teams to have visibility into who committed changes, on what date, and view notes surrounding the changes for additional context. FTP/SFTP by contrast is like the wild west - anyone can make changes on the fly without others knowing, and without the context letting your team know why things changed.

Git is widely understood to be the best way to version control your website's files. WP Engine offers git built into the platform already, which is a great way to start. However, most teams will also want to collaborate on an external resource like Github, Gitlab, or Bitbucket. This allows teams to integrate into the CI/CD tools they need to automate their builds, tests, and deployments in combination with WP Engine.

We have published guides on integrating Github, Gitlab, and Bitbucket with a CI/CD system with WP Engine. The majority of our users prefer to use DeployBot for full website version control, and CodeShip for more branched solutions (individual plugin or theme management).

Local dev



When using version control as a team, each developer "checks out" pieces of code to edit, then commit back to the common repository. Where does the code go when it's checked out? To each developer's local development environment. Developers should copy the codebase down to their local development instance, make the changes needed, then commit the changes back to the repository using git. For the most simplistic solution, you can use MAMP, or Desktop Server. Both of these solutions present a low barrier of entry, and if you're just getting started with local development, this will help you get to the development part faster, rather than spending valuable time configuring your local environment. For more tailor-made solutions, read on.

VAGRANT

Vagrant is a local development tool that aims to mimic production environments as closely as possible. It works with any operating system on your local computer. Developers can initiate a vagrant using command line, and simply pull their website files down using git.

VIRTUALBOX

On top of Vagrant, many developers also use VirtualBox. VirtualBox is a virtualization tool used to run different operating systems side-by-side. For example, some Mac users utilize the Parallels program to virtualize a Windows environment on their Mac machine. In a similar way, VirtualBox allows users to run a Linux/Ubuntu machine from their native operating system, to mimic the production environment where their website is actually hosted.

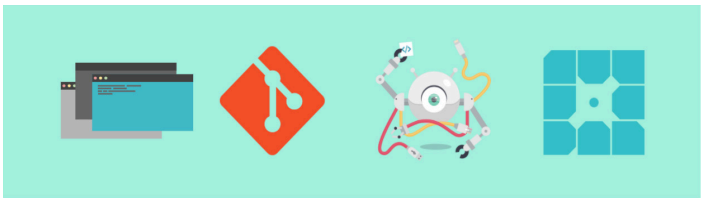
VVV

VVV, or Varying Vagrant Vagrants, is a software created by 10up to take your vagrant solution even further. VVV is installed on top of VirtualBox and Vagrant on your local machine, and offers a WordPress-specific vagrant system. It even offers local use of the powerful WP CLI toolset. Due to its WordPress-focused environment, it is an ideal solution for developers wanting to contribute a theme or a plugin to the WordPress.org repository.

CI/CD

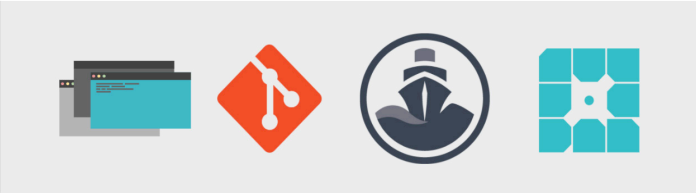
Continuous Integration (CI) and Continuous Deployment (CD) are two workflows baked into Agile development. In order to create rapid, working iterations, it's important to commit early, commit often, and deploy frequently. Your deployments will often be set at a cadence of one or two weeks, or whatever "Time Box" you have decided upon for your team's Sprint cycles. Integration, or committing of code should happen daily for all developers. This means less wasteful code development--each piece is tested to ensure it meets quality standards, so errors are caught at the beginning. So what tools allow for a CI/CD pipeline? We will go over some of our favorites below.

DEPLOYBOT



DeployBot is one of the more simple CI/CD tools to use when it comes to website development. It simply polls your Github or Bitbucket repository for changes frequently, then pushes those changes to WP Engine (ideally your Development environment for your WP Engine Site) using SFTP. With DeployBot your developers each have a pipeline from local development, git push to their SCM (Github or Bitbucket), and DeployBot handles any builds and the deployment to WP Engine. DeployBot is ideal for users who manage their full site in a single branch, and not segmented further than that. Its simplicity is what draws many users to this CI/CD pipeline.

CODESHIP



CodeShip offers a more powerful toolset, allowing for all kinds of software development, testing, build, and deployment integrations. CodeShip supports Github, Bitbucket, and Gitlab as SCMs which they poll for changes regularly. What it does after detecting a change is up to you and the needs of your team. You can hook into Slack channels, hosting services, vagrants, and more. You can test, then run builds against any dependencies needed. And most importantly, you can then deploy via git to your Development environment on WP Engine using a custom script. Since you can deploy with a custom script, this also allows developers the freedom to manage and push multiple branches using environment labels. CodeShip is easy enough to setup that any level of developer will find success, but offers a more robust feature system that businesses of all levels will be able to tailor a solution to their own needs.

JENKINS

Jenkins is a commonly used CI/CD tool for almost all software development. Our own R&D team at WP Engine uses Jenkins to help automate regular deployments and test builds. Jenkins can hook into a wide range of tools including your chat systems and monitoring to help increase awareness, so it is highly extensible. Jenkins comes with a learning curve, though. While it is fairly easy to install, it is not so easy to configure your deployment pipeline, which happens with a custom script via SSH Gateway. Jenkins is a powerful tool that should be used by larger teams of experienced developers. While there is an abundance of online documentation, it can be difficult to digest if you're just getting started.

Build tools

In the development process, there is a good chance you'll encounter files which need to be built or compressed as a final step--most often, JavaScript files. A common example with JavaScript development is using Webpack to do your final build--this build compresses and bundles up your JavaScript files, which means they aren't easily editable once they are deployed to Development, Staging, or Production. It is best practice to Build your files using the CI/CD service you utilize. Keep in mind, if your Build has dependencies, these should be ignored using the .gitignore file so they are not pushed to production. Grunt and Gulp tend to be the "task runner" build tools most JavaScript developers use, while Webpack is more of a "bundler" tool. Which you choose really comes down to personal preference, experience level, and project needs (configuring structure, or strictly coding).

Code testing

Not only is it important to regularly commit and deploy working products, those products should also be held to a strict standard of Unit Testing, Acceptance Testing, and Usability Testing to ensure the best experience for your end user. Remember, Agile methodologies center around products that fulfill customer needs. Your customer deserves the best product. So what kind of automated tests should you run? We'll dive in below.

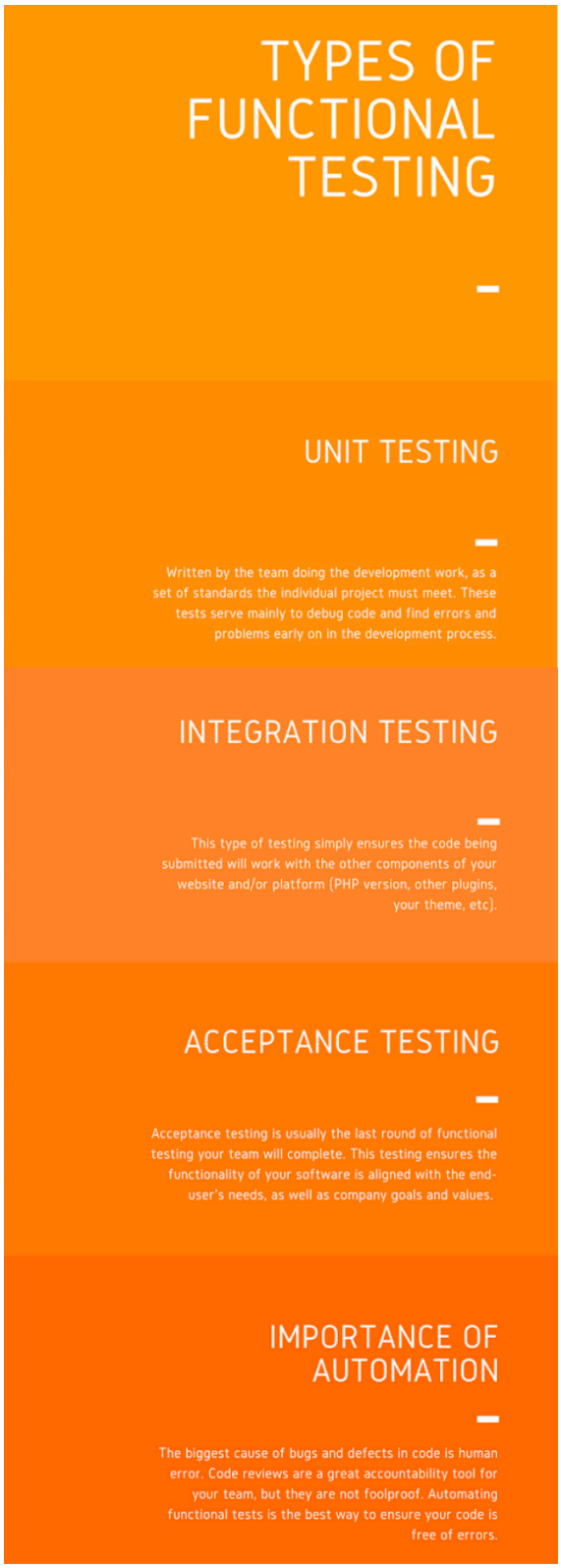
UNIT TESTING

Unit tests are often written by the team that tackles a given project, before any development on the project has begun. These tests will look different for any team and any project, but generally these tests just serve to identify and help developers debug code at an early stage. It is highly encouraged to automate these tests when code is committed back to the common repository.

INTEGRATION TESTING

Integration testing is fairly straightforward: it simply ensures that the code you've written "plays nice" with the other pieces of the website. For example, if you've modified your theme's functions.php file, your integration testing would verify that your theme is still compatible with your site's plugins and WordPress core as

a whole. Integration testing should also be performed regularly, prior to Sprint releases.



ACCEPTANCE TESTING

Acceptance testing is the last round of functional tests which should be required by your Agile team. Acceptance testing holds the end product against business and company standards, as well as the end-user requirements to ensure they are aligned. This step often requires internal and external user engagement with the product, which often is labeled as a "Beta" or "Release Candidate" when presented to users. This phase is only needed once the product as a whole is nearly ready for market.

TEST-DRIVEN DEVELOPMENT WITH CODECEPTION

In an ideal world, your team should use the following workflow with testing: write a small unit test first, then write your code until your small unit test passes. The next time you add functionality, write another small unit test to add to the existing one before coding just enough for this test to pass. In this way, your coding and unit tests are scalable while ensuring your existing code does not break with new deploys. Codeception for WordPress is a tool that has pre-built integration, functional, and acceptance tests for WordPress that test against WordPress defined functions and classes. When using this tool, developers can more easily adhere to WordPress coding standards.

Monitoring

Once your website code has been deployed to your Production website, you can utilize monitoring on several vectors to ensure your product remains operational at all times.

SERVER MONITORING

Monitoring the health of your server ecosystem is important, and you can use services like Zabbix and PagerDuty to monitor and alert your team if your system is overloaded or otherwise non-operational. If you host your websites on WP Engine, we take care of server-level monitoring on your behalf, and work to resolve any issues often before your team is ever aware there of it.

UPTIME MONITORING

WP Engine takes care of ensuring your web server's uptime, but there are other vectors to be aware of as well. For example, if the code you just deployed causes a fatal error, uptime monitoring services will catch this almost instantly so your team can either fix or roll back the change. Pingdom and UptimeRobot are two uptime monitoring services you can use for this purpose.

APPLICATION-LEVEL PERFORMANCE MONITORING

For WordPress, Application-Level Monitoring will give you insight into the speed of your "PHP Application," or WordPress website. It will separate performance into PHP, Database queries, and external calls, and help you drill down into specific code performance issues. At WP Engine, we partner with New Relic for the best insights into performance. You can purchase Application Performance through WP Engine to get these insights from New Relic to troubleshoot and improve your website performance.

FILE INTEGRITY MONITORING

Last, it's important to be aware of file changes on your website for security purposes. When working on a team of Agile developers, everyone should be aware of who is making changes, and when. While git gives you the ability to add comments and a running log of changes, file changes via SFTP do not. However, WordPress plugins like Stream and Sucuri Security will help give your team this extra visibility.

Copy to/from environments

Another challenge development teams face is how to copy between Development, Staging, and Production environments. On WP Engine, this is easy with our Copy To/From tool within each Site, which utilizes our automated backup system. On other hosts, you might have to utilize other migration tools or a 3rd party deployment system to deploy changes between environments.

Migration

If you or your team have ever had to handle a website migration, you know it can get complicated quickly. Moving all your files, database, and changing DNS can be a struggle for even the most advanced developers. WP Engine takes the hassle of site migration and bundles into a simple WordPress migration plugin, so you can migrate your website with the click of a button.

If you do not use WP Engine, there are other backup and site clone tools that will help you accomplish migration in a few relatively easy steps. Some commonly used plugins are: Duplicator, All-in-One WP Migration, and BackupBuddy.

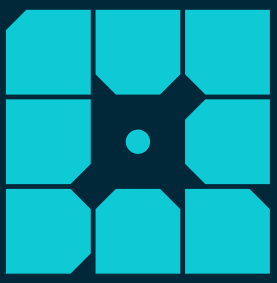
Review.

Agile development is often referred to as the "right" way to develop as a team, and for good reason. Not only does it prevent burnout and increase team morale, it also opens visibility and communication for your stakeholders and allows for rapid iteration of new features. It doesn't take much to see the inherent value for all parties involved: your end-user, your team, and your stakeholders. Inevitably, the path to Agile will be clumsy and confusing at first, but the end result well worth the any stumbles along the way. Those who have used Agile methods come away as strong believers because of how efficient, manageable, and visible their projects become. To get started, gather your team in a room and hash out your daily tasks and calendars. Find where there is wasted time, and resolve to fix it in your first, second, or third Sprints. In time what once was arduous and manual work will come more easily and faster, and your website's end users will be delighted by it.

About the author.

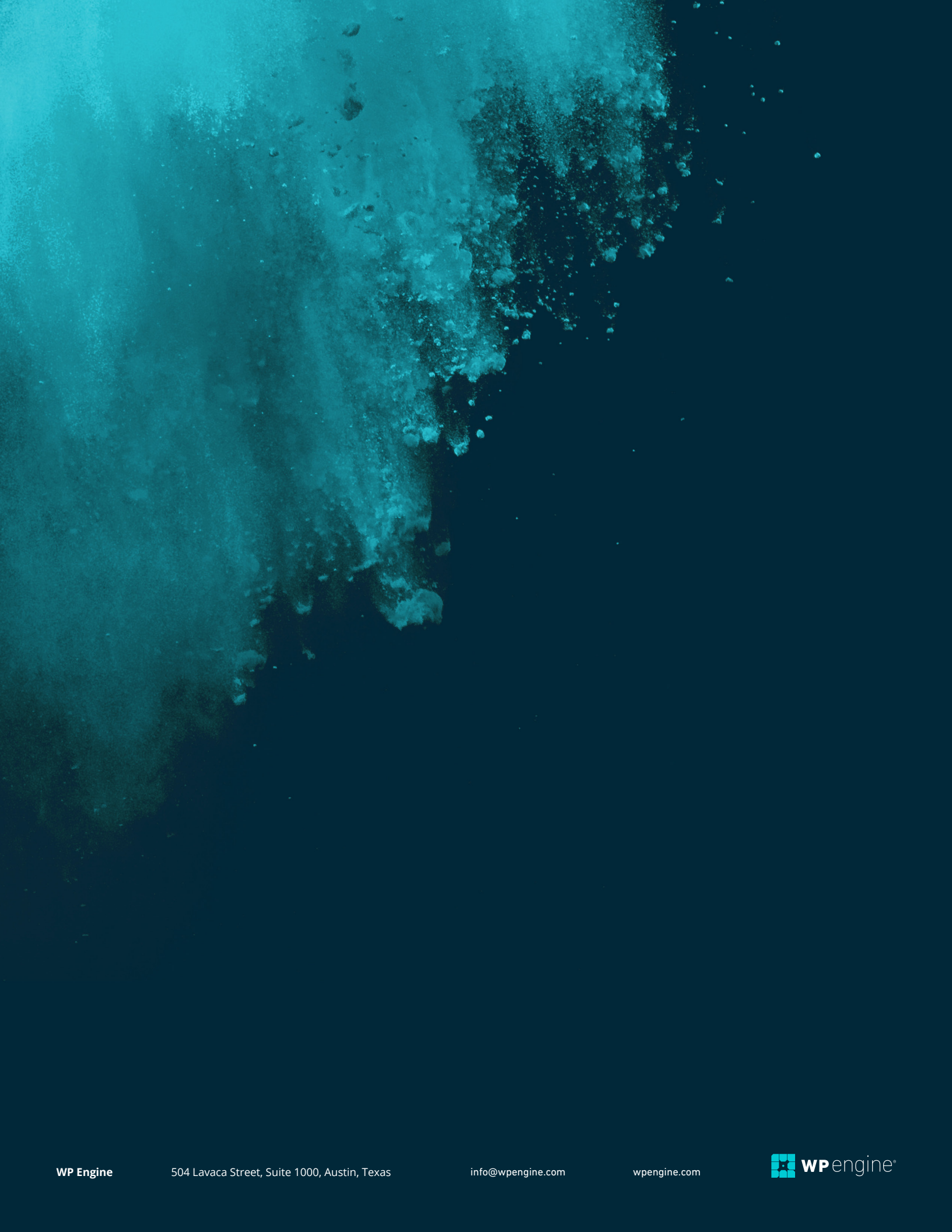
Janna Hilferty

Janna Hilferty is a CX Operations Content Specialist at WP Engine. She loves both technical and free-form writing, hiking with her dog, and painting with all the colors of the wind. In her free time you can find her blogging, smoking a cigar, or watching cheesy documentaries.



About WP Engine.

WP Engine is the world's leading WordPress digital experience platform that gives companies of all sizes the agility, performance, intelligence, and integrations they need to drive their business forward faster. WP Engine's combination of tech innovation and an award-winning team of WordPress experts are trusted by over 70,000 companies across 130 countries to provide counsel and support, helping brands create world-class digital experiences. Founded in 2010, WP Engine is headquartered in Austin, Texas, and has offices in San Francisco, California; San Antonio, Texas; London, England; Limerick, Ireland, and Brisbane, Australia.
www.wpengine.com



WP Engine 504 Lavaca Street, Suite 1000, Austin, Texas info@wpengine.com wpengine.com

WP engine®