

A top-down view of a developer's workspace on a wooden desk. A laptop is open, displaying a web browser with a 'WP Engine DevKit' interface. The browser shows a 'Keywords Tool' for contributors, with a code editor displaying HTML and CSS. A hand is visible typing on the laptop keyboard. To the left of the laptop, a tablet displays code. A pair of glasses, a glass of amber liquid, and several pens are scattered on the desk. The overall scene is dimly lit, with a focus on the laptop screen and the text overlay.

# Maximizing your development workflow with the WP Engine DevKit.



WP engine®



DEVKIT



Table of Contents.

Overview: A developer workflow.....4

SETUP .....5

BUILD.....8

LAUNCH .....9

MAINTAIN.....9

Building with the WP Engine DevKit.....12

WORDPRESS SITE MANAGEMENT .....13

DATABASE MANAGEMENT .....14

DEMOS, PREVIEWS, AND “LIVE” TESTING .....14

CACHING .....15

WEB TRAFFIC TESTING.....15

SMTP TESTING.....16

AUTOMATED TESTING .....16

DEBUGGING .....17

OTHER TOOLS .....17

Feature Roadmap .....19

WP ENGINE API.....20

THE GENESIS FRAMEWORK AND STUDIOPRESS THEMES.....21

ATOMIC BLOCKS .....22

Support & Feedback .....23

GOING FURTHER (RESOURCES).....24

About DevKit .....25

About WP Engine.....26

# Maximizing your development workflow with the WP Engine DevKit.

You'll hear the phrase "development workflow" tossed around quite a bit in development communities. What exactly does it mean?

A development workflow is an established set of tools, systems, and processes used by either a single developer or a team of developers throughout the lifecycle of a web project. To put it simply, a development workflow is a consistent way of doing all the things that fall under the umbrella of web development, starting with initial project setup all the way through project launch and ongoing maintenance.

A good development workflow brings many benefits to a web project including organization, consistency, and a refined process. This translates directly into more effective (and efficient) project cycles, bringing down your total time and cost of development for internal or client projects.

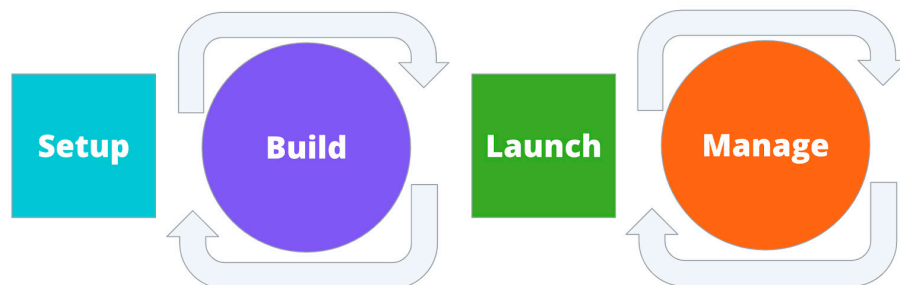
**In this ebook, we'll look at an overview of a web development workflow with a specific focus on the Build phase. After reading this, you'll understand conceptually what a modern development workflow entails and how you can leverage tools built into the WP Engine Digital Experience Platform to streamline your systems.**





# Overview: A developer workflow.





When it comes to establishing a development workflow, there are a variety of tools and services at your disposal. Developers vary in their preferences, so every workflow will look a little bit different depending on the exact combination of resources used.

In the following pages, we'll outline the major steps involved in any modern web project and make reference to some of our favorite open source tools. We'll also discuss bonus tools built into WP Engine's Digital Experience Platform you can use to create faster, simpler routines for your WP Engine-hosted sites.

## Setup.

Once project requirements are agreed upon by stakeholders and development is given the "green light," it's time to set up the foundation for a web project.

### Repository setup.

If you and/or your team uses version control, the first step is to set up a repository. Git is by far the most popular, however some developers prefer Subversion (SVN).

Regardless of your preferred method, version control is a critical component of professional web development, especially if you're working on code with multiple contributors.

Even though WordPress core development uses SVN, most developers find it simpler to start with Git.

For more information about getting started with version control, check out [this article](#).

### Version control hosting platform.

Most of the time, you'll want a platform to host the version control system. This is especially true if you're working with a team—you'll need your version-controlled code hosted where everyone on the team has access to it. Popular platforms include Github, Bitbucket, and GitLab.

### Local Dev Environment.

All development should happen locally prior to being uploaded (or pushed) to a live server. Local development includes a number of practical benefits including speed of environment and the ability to perform development work without an Internet connection, not to mention that when developing locally you're not in danger of messing up anything on a live site.

Version control is a critical component of professional web development.



*SSH Gateway is ideal because it provides developers an easy, secure interface to interact with a site's files and content.*

All you need to run WordPress locally is PHP and MySQL (or MariaDB). Traditionally you could set up these environments using tools such as MAMP, WAMP, and XAMPP (based on your operating system).

Some developers prefer using virtual machines or virtual environments. There are many options available, each with their own strengths, but in this document we'll focus on **the WP Engine DevKit**, a suite of tools that includes a [Docker](#)-based solution for local development you can use regardless of your hosting provider (though you can enjoy advanced integrations if you're hosting with WP Engine).

#### **SSH Gateway.**

SSH stands for "Secure Shell" and is the authentication method two devices (computers, servers, or other Internet devices) use to communicate with each other.

On the WP Engine platform, SSH Gateway access means the ability for you to connect remotely from your local machine to a container where your site's content is hosted.

SSH Gateway is ideal because it provides developers an easy, secure interface to interact with a site's files and content. Many Integrated Developer Environments (IDEs) and code editors give you the ability to configure SSH settings directly within the editor, meaning you can interact with remote files without the need for an S/FTP client to move files between servers.

#### **WP Engine API.**

As the name suggests, the WP Engine API is unique to WP Engine and only available to customers. The WP Engine API allows you to interact with the platform programmatically.

Use cases include:

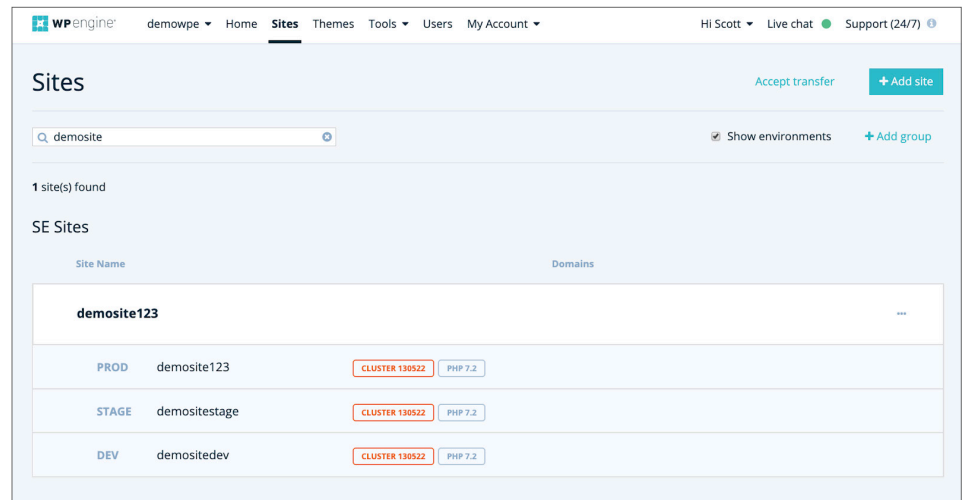
- Combine with WP-CLI for automatically creating and configuring sites
- Assist in batch migrations
- Integrate WP Engine as part of your product offering.

For more information, check out this webinar: [The WP Engine Developer Experience – Increased agility, improved efficiency.](#)

When ready, go here to [enable the WP Engine API](#) in your WP Engine account.

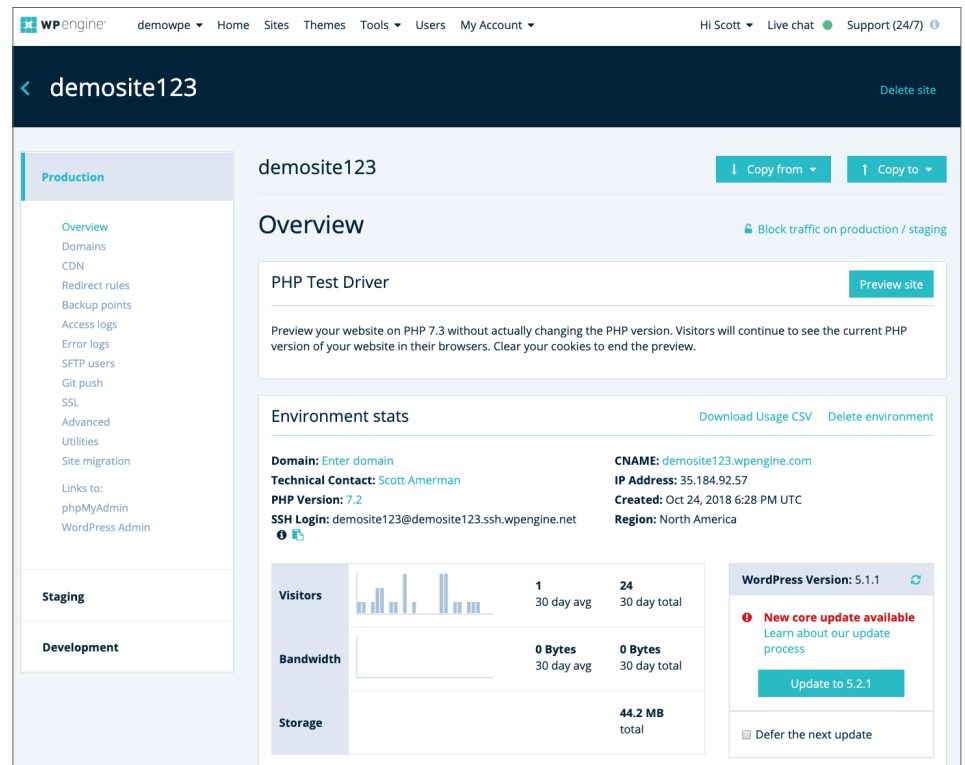


## User Portal.



Clunky cPannels are for the old days. WP Engine streamlines the way you manage multiple aspects of site set up through its User Portal. It's a single location where you can manage:

- Site creation
- Site configuration
- User accounts
- User activity logs
- Site environments
- Establish SSH and Git access
- SSL
- Lots more...



Click here to [explore the new User Portal](#) experience.



*Nobody wants to recreate the wheel with every project. That's a waste of time and fails to take advantage of the benefits that come with repeated exposure to the same code base.*

## Build.

We'll cover the Build phase of a project in greater detail shortly. Here's a highlight of some of the elements included in a development workflow.

### Collaborative workflows with Git.

Using [Git](#) you can streamline your workflow, especially when working with a team. Since Git allows for notes, pulling down changes pushed by other developers, and unique IDs for every push, it's easy to determine what change was made by who, when, and for what reason.

Additionally, git is ideal when working with large sites. Only the files you choose are pushed, which shortens your deploy process when compared to a full site copy.

### Starting templates.

Nobody wants to recreate the wheel with every project. That's a waste of time and it fails to take advantage of the benefits that come with repeated exposure to the same code base.

Many developers start with [\\_s](#), [WP Rig](#), the [Genesis Framework](#), or their own custom starter template.

WP Engine customers have the Genesis Framework + the entire suite of [StudioPress Themes](#) at their disposal to use as a starting point for both custom and turnkey projects. Genesis is also tightly integrated with [Atomic Blocks](#) which gives developers access to page-building blocks such as a Call to Action block, a Post Grid block, and an Inline Notice block.

With minimal effort, developers can quickly apply custom components to any StudioPress Theme or child theme made for the Genesis Framework.

Once you get your base foundation for a web project (theme, core plugins, etc), you can use WP Engine's [Copy to/Copy from](#) feature to replicate that environment over and over as the starting point for new projects.

Additionally, DevKit contains a clone tool enabling you to quickly clone a WP Engine site environment to your local environment via command line.

### Include dependencies.

Depending on the complexity of a project, you may need to include additional code libraries as dependencies in your project. [Composer](#) is a popular tool for dependency management in PHP. With it, you can declare the libraries your project depends on and it will manage (install/update) them for you.

If you're not familiar with using Composer in a WordPress project, [here's a good primer](#).

## Testing.

Testing is an important part of any build process. It helps ensure your code is error free, follows WordPress coding standards, and works as expected.

*These tools allow multiple developers to check out the code they wish to edit, make changes, merge it back into a common repository, and automate tests against the changes.*

[PHPUnit](#) is the official PHP testing framework used by WordPress core. We'll discuss it along with other testing tools later in this document.

Additionally, if you want to demo a new version of PHP on a WP Engine site, you can use the [PHP Test Driver](#).

## Launch.

Launch sequence varies by developer and the complexity of a project, but it typically involves the following:

- Configure site (SSL, CDN, etc)
- Configure SEO & redirects
- Configure site analytics
- Quality assurance testing
- Review

While this could be done in as few as two environments (i.e. Staging and Production), WP Engine provides multiple environment types you can use to move a site through a launch sequence. Below are the environment names, abbreviations, and our recommendations for how to use them:

- **Development (DEV)** – Primary environments for building and developing your site
- **Staging (STAGE)** – Pre-production environments that can be used to test code before deploying to production
- **Production (PROD)** – Environments that are production-ready and (usually) publicly facing.

## Maintain.

Even after a site is launched, it still requires attention. Considering how you will maintain a site post-launch is an important part of a development workflow.

### Continuous Integration / Continuous Deployment.

If you work with a team of developers, chances are you want to use Continuous Integration (CI) and/or Continuous Deployment (CD) tools. These tools allow multiple developers to check out the code they wish to edit, make changes, merge it back into a common repository, and automate tests against the changes. Additionally, some CI/CD tools also allow for automated deployments when tests are successful, or at predetermined intervals.

There are various tools available for CI/CD. Here are detailed instructions for [using Codeship with WP Engine](#) or [DeployBot with WP Engine](#).

### Application Performance Monitoring.

WP Engine offers Application Performance Monitoring (APM) as an add-on to certain plans. We've partnered with [New Relic](#), leaders in digital intelligence, to provide APM as part of our intelligence suite.



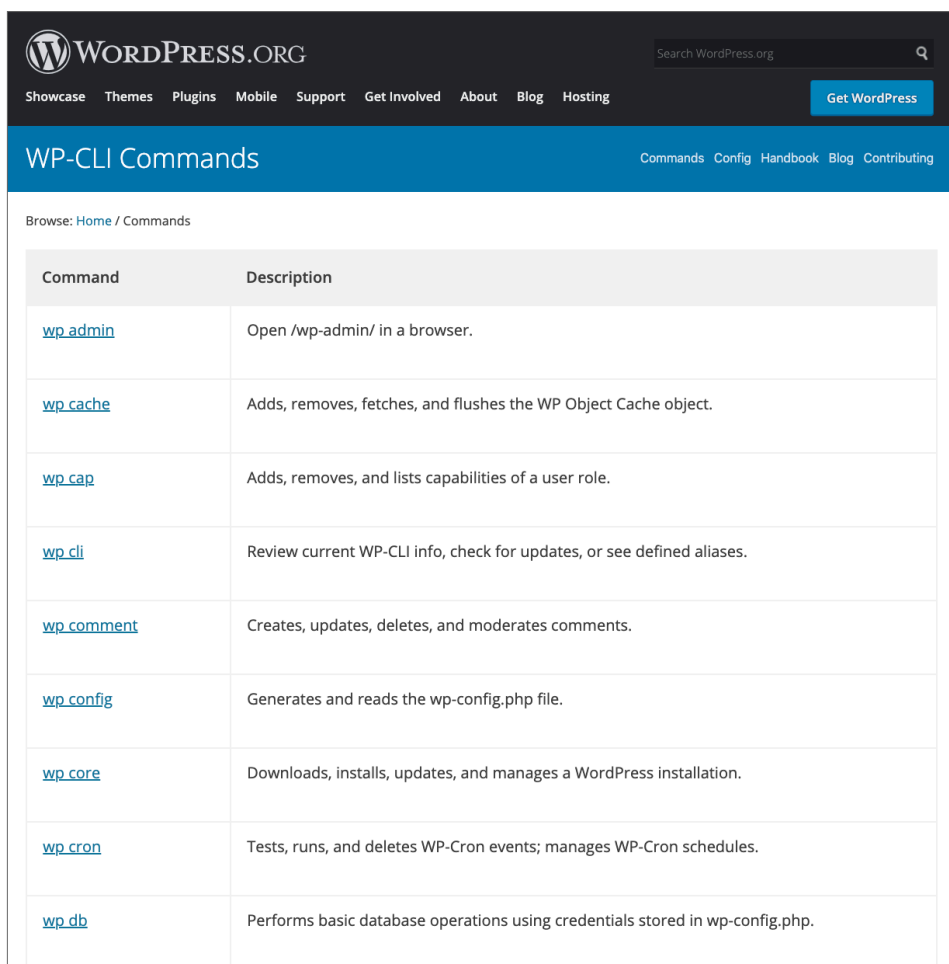
*With WP-CLI, you can perform common maintenance tasks across individual sites or all sites you manage, saving you time and reducing repetitive tasks.*

Application Performance is a tool built to help you combat application and website bottlenecks so you can provide an optimized WordPress experience. This tool provides code-level visibility to help teams troubleshoot faster, optimize their WordPress experiences, and ultimately deliver a better user experience. You can use APM to discover the root cause of any site issues by drilling down into slow queries, transaction traces, errors, and more.

Learn more about how you can use APM to optimize your entire technology stack with detailed performance metrics via this [white paper](#) or [webinar](#).

### WP-CLI for basic maintenance.

Performing basic site maintenance is routine, dull (let's be honest), and consumes more time than it should.



The screenshot shows the WordPress.org website with a dark header. The main navigation bar includes links for Showcase, Themes, Plugins, Mobile, Support, Get Involved, About, Blog, and Hosting. A search bar is on the right. Below the navigation bar is a blue banner for 'WP-CLI Commands' with links to Commands, Config, Handbook, Blog, and Contributing. The main content area has a breadcrumb 'Browse: Home / Commands' and a table of WP-CLI commands.

Command	Description
<a href="#">wp admin</a>	Open /wp-admin/ in a browser.
<a href="#">wp cache</a>	Adds, removes, fetches, and flushes the WP Object Cache object.
<a href="#">wp cap</a>	Adds, removes, and lists capabilities of a user role.
<a href="#">wp cli</a>	Review current WP-CLI info, check for updates, or see defined aliases.
<a href="#">wp comment</a>	Creates, updates, deletes, and moderates comments.
<a href="#">wp config</a>	Generates and reads the wp-config.php file.
<a href="#">wp core</a>	Downloads, installs, updates, and manages a WordPress installation.
<a href="#">wp cron</a>	Tests, runs, and deletes WP-Cron events; manages WP-Cron schedules.
<a href="#">wp db</a>	Performs basic database operations using credentials stored in wp-config.php.

With WP-CLI, you can perform common maintenance tasks such as comment moderation, plugin updates, and user permissions across individual sites or all sites you manage, saving you time and reducing repetitive tasks.

Learn more about using WP-CLI with WP Engine in this [webinar](#).

Every developer varies in how they like to work, and their individual workflow will reflect these differences.

### Tools from the User Portal.

Your WP Engine User Portal provides a number of site management tools and utilities you can use as part of your maintenance workflow.

Here is a sampling of what you can do:

[Manage account users](#)

[Redirect management](#)


[Advanced tools such as a read-only filesystem and cache exclusions](#)

[Troubleshoot issues with access logs and error logs](#)

[Access to backup points and 1-click site restoration](#)

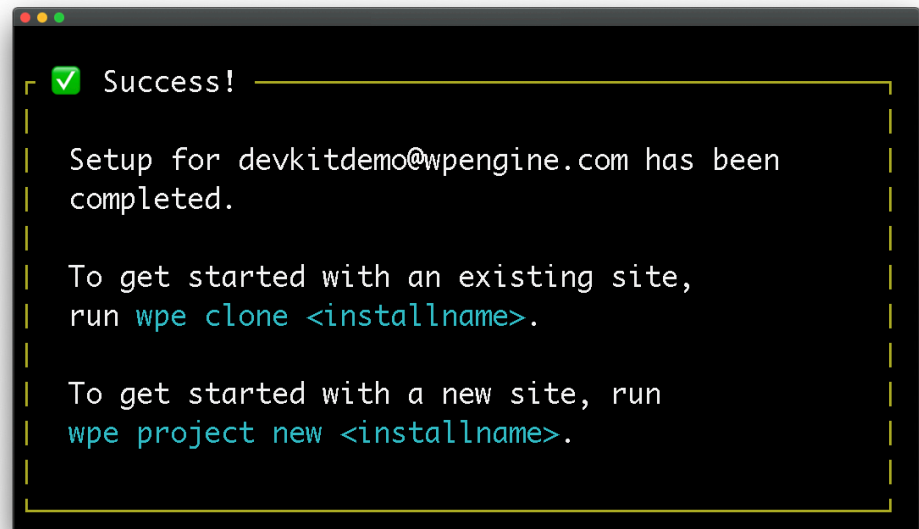
To learn more, explore [what's available in your User Portal](#).





# Building with the WP Engine DevKit.

*DevKit is a suite of tightly integrated tools designed to take your workflow to the next level.*



In this section, we'll focus on the WP Engine DevKit, a suite of tightly integrated tools designed to take your workflow to the next level in terms of both proficiency and professionalism.

At the heart of DevKit is a Docker-based environment for building WordPress sites locally. As mentioned earlier, there are a number of available solutions for creating a local development environment, however DevKit comes with some notable advantages:

- Quickly connect to your remote server via SSH with a simple command:

```
> wpe ssh
```

Once on the server gateway, you can run commands just as if you were on the server.

- Quickly connect to your local Docker container using the command:

```
> wpe bash
```

- DevKit includes various build tools you can run inside your local environment, which means you don't have to worry about having them all installed and configured properly on your computer including:

• wp (WP-CLI)	• Varnish	• PHPunit
• ngrok	• Nginx	• xdebug
• phpMyAdmin	• Mitmproxy	• webgrind
• memcached	• Mailhog	• mySQL

Assuming you've got your local development environment running, let's take a look at each of these dev tools. If you have not set up your local development environment, please refer to [the WP Engine DevKit Documentation](#).

## WordPress Site Management.

Regardless of the stage you're at in the development of a WordPress project, site management will always be a central component. Whether it's managing plugins, user permissions, or testing against certain versions of WordPress, you'll need to manipulate details about a site.



Traditionally, this is accomplished by logging into the WordPress admin, but there's a much more efficient way...

### WP-CLI.

WP-CLI is the command-line interface for WordPress. You can use it to update plugins, configure multisite installs and much more, without using a web browser. Additionally, you can use WP-CLI to manage multiple WordPress sites on your WP Engine account with single commands, saving you the time of logging in to individual dashboards to perform basic management tasks.

While you can access a command line window and use WP-CLI via the “Advanced Tools” tab in your WP Engine User Portal, there are some [known limitations](#). We recommend you **utilize WP-CLI directly via DevKit (to access your local environment) or via SSH (to access your hosted server)**. This allows you to fully use WP-CLI to manage your WordPress site(s).

To learn more, check out this free webinar on [how you can use WP-CLI with WP Engine](#). For a full list of capabilities and commands supported by WP-CLI, here is the [official documentation](#).

## Database management.

With WP Engine, you can access and manage site databases directly via phpMyAdmin. For the hosted version of the site, you can access phpMyAdmin from the site environment page in your User Portal. Alternatively, you can interact with the hosted database via SSH and command line with the following command:

```
> wpe ssh mysql
```

This connects you to your hosted environment via SSH and brings up a mysql prompt in terminal. If you're not familiar with interacting with a database this way, here's a [cheat sheet of mysql commands](#).

To access phpMyAdmin for your local database, use this command:

```
> wpe alpha tools phpmyadmin
```

This launches phpMyAdmin in a browser window at a URL that follows this pattern (be sure to substitute myinstallname with the name of your local install): `http://phpmyadmin.myinstallname.wpengine.test/`

Alternatively you can manually export the data from your hosted database and manually import into your local database. This process can be reversed if you want to replace a hosted database with a copy of your local database.

## Demos, previews, and “live” testing.

There will inevitably be times when you'd like to show a feature or preview of a site to someone outside of your local network. There will also be times when you want to test an aspect of a site on an actual device versus a simulated device or in a particular environment.

WP-CLI can help you work more efficiently so you can focus on the projects you want to.

No need to push up to a staging environment. Use ngrok!

**Ngrok** is a tool you can use to expose a local server to the public Internet by assigning a randomized URL. If you're using DevKit and would like to make your site securely available on the public web, you can start the ngrok service and display the site URL with the following commands:

```
> wpe start ngrok
```

```
> wpe ngrok --print-url
```

If you'd like to open the ngrok url in your local browser, use:

```
> wpe ngrok
```

Note that, unlike some other tools included with DevKit, ngrok does not automatically start as part of the `wpe start` command. This is because ngrok has an eight hour session limit and there's no need to start ngrok until you're ready to use it.

## Caching.

The more closely your development environment matches a production environment, the more effectively you can test a website.

WP Engine's caching solution includes [memcached](#) and [Varnish](#). Memcached provides object caching PHP code for faster subsequent run times, while Varnish provides full-page caching.

Memcached is started by default when running `wpe start`.

Varnish is not started by default when running `wpe start`. Full-page caching is typically not part of a daily development workflow due to the frequent code changes, but it's helpful for debugging any caching issues that might occur in a hosted environment.

To start Varnish locally and view a cached version of the site, start by running this command:

```
> wpe alpha tools varnish
```

That command opens a browser window with the cached version of your local site. The URL follows this pattern (be sure to substitute `myinstallname` with the name of your local install):  
`http://cached.myinstallname.wpenginesite.test/`

To toggle back to an uncached version of your site, use this URL structure:  
`http://myinstallname.wpenginesite.test/`

## Web Traffic Testing.

DevKit is also bundled with [mitmproxy](#), a tool that captures web traffic requests. It also blocks all post requests, meaning that no plugin you're using can trigger an outgoing request. For example, if you're using Jetpack or another social sharing plugin to test if it auto-posts to Twitter when publishing a new post, mitmproxy will stop that traffic, but you will be able to confirm whether the request was generated successfully.

*The more closely your development environment matches a production environment, the more effectively you can test a website.*



Mitmproxy is your swiss-army knife for debugging, testing, privacy measurements, and penetration testing.

To view all web traffic with mitmproxy, run the following command:

```
> wpe alpha tools proxy
```

This launches a browser window for mitmproxy that captures traffic requests as you test your site.

Visit the [official mitmproxy documentation](#) for the various ways you can use mitmproxy to analyze http and https requests.

## SMTP Testing.

There will be times when your WordPress site triggers an email. It could be an admin notification, a password reset, a form submission, or a number of other possibilities.

In a development environment, you want to know that emails are being appropriately triggered, but you don't want them to actually send.

[MailHog](#) is an email testing tool you can use for SMTP delivery. It enables you to:

- View messages in the web UI, or retrieve them with the JSON API
- Optionally release messages to real SMTP servers for delivery

DevKit comes with built-in Mailhog support to capture all outgoing emails—no configuration necessary! You can view the Mailhog interface by running:

```
> wpe alpha tools mail
```

## Automated Testing.

A critical part of any professional development workflow is code testing and quality assurance. You want to make sure that the software you've created works as expected under a variety of use cases.

At a basic level, this starts with linting your code (there are a variety of [linting packages](#) available if you want to automate this process with Grunt or Gulp).

If you're using WP-CLI, you can additionally use [wp-cli-lint](#) to ensure your code meets [WordPress core coding standards](#). If you're not using WP-CLI, you can alternatively use [PHP\\_Codesniffer](#) to comply with core standards. (Here are instructions for [using PHP\\_Codesniffer with WordPress](#).)

### Phpunit.

Testing code is an important part of a professional development workflow. It's time consuming (and not practical) to manually test every part of your custom WordPress build every time you update your code. That's where automated testing comes into the picture.

PHPUnit is the official testing framework used by WordPress core to test PHP code. You'll also hear this referred to as unit testing. You'll want to write tests according to your custom code.

*The most important part of any test is the **assertion**. An assertion is a comparison between the value you expect to get from the system, and the value that you actually get. The very simplest tests may consist of nothing but a single assertion.*

- [WordPress Core Handbook on PHPUnit tests](#)

To use PHPUnit with DevKit, you'll first want to bash in to your local install using this command:

```
> wpe bash
```

Once that's done, you see a list of available PHPUnit options using this command:

```
> phpunit --help
```

You can find more information on working with PHPUnit and writing PHPUnit tests in the official [WordPress documentation](#).

## Debugging.

### Xdebug.

[Xdebug](#) is a PHP extension for debugging code. It enables you to “step” through code at breakpoints you determine to see exactly what's happening in your code as it processes.

Xdebug works with most IDEs, including Sublime Text, PHPStorm, Atom, and VS Code. You can start debugging from within your IDE (if you have an xdebug package installed). Alternatively, with DevKit you can start it directly from the command line with this command:

```
> wpe alpha xdebug
```

### Webgrind.

While Xdebug can output information within your code environment, sometimes it's helpful to get a large-scale overview of application performance. That's where Webgrind comes in.

[Webgrind](#) is a frontend viewer for xdebug output and, because it's included in DevKit, no additional configuration is required to use it. Run the following command to launch Webgrind.

```
> wpe alpha tools webgrind
```

## Other tools.

There are some other useful command line tools provided by DevKit worth a mention.

### Launch wp-admin.

Keeping track of your admin name and password for various sites can be a pain. DevKit provides a simple way to automatically launch wp-admin.

```
> wpe login
```

**Running WP-CLI commands locally.**

There are a couple of ways you can work with WP-CLI locally. The first is to use this command to bash into your Docker container:

```
> wpe bash
```

Once there, you can run WP-CLI commands just as you would expect. For example, you could list all local plugins using:

```
> wp plugin list
```

Alternatively (and what we recommend), with Devkit you can shortcut this with a single command:

```
> wpe wp plugin list
```

In this case, wpe simply serves as a wrapper for the bash command to give you a faster way to use WP-CLI.

**Additional commands.**

For a full list of commands supported by DevKit, simply run this command in terminal:

```
> wpe --help
```

DevKit is under active development and will continue to make best-in-class tool integrations available to developers. As you explore DevKit, be sure to reference the [official documentation](#) for installation, usage, and troubleshooting information.



**Feature  
roadmap.**



*When you partner with WP Engine, you can focus on your core strengths while leveraging us to evaluate the “shiny new” development tools.*

At WP Engine, one of the ways we help you win online is by partnering with you for technology insights. The development landscape is ever-evolving (and at a rapid pace, to boot!). Our team of engineers continues to monitor trends, tools, and systems and then works to integrate best of class technologies into the WP Engine Platform.

That means that when you partner with WP Engine, you can focus on your core strengths while leveraging us to evaluate the “shiny new” development tools. We’ll integrate the best ones into our platform so you can take advantage of them.

With that in mind, let’s look at what you can look forward to with WP Engine in the near-term.

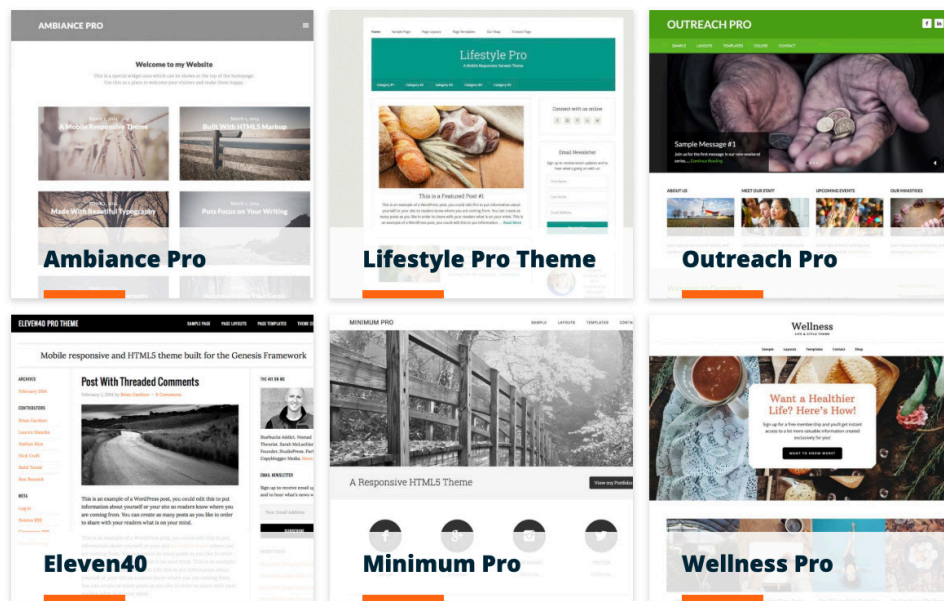
## WP Engine API

The WP Engine API allows you the ability to interact with our platform programmatically. The API currently supports the following list.

- Accounts
  - List all WP Engine accounts
  - List account by ID
- Sites
  - List all sites
  - Create a new site
  - Get a site by ID
  - Change a site name
  - Delete a site
- Installs
  - List your WordPress installations
  - Create a new WordPress install
  - Get an install by ID
  - Delete an install
  - Update a WordPress install
- Domain
  - Get the domains for an install by install ID
  - Add a new domain to an existing install
  - Get a specific domain for an install
  - Set an existing domain as primary
  - Delete a specific domain for an install
- User
  - Get the current user

You can enable the WP Engine API in your account by [following these instructions](#).

## Genesis Framework + StudioPress Themes.



*We'll continue to help you produce beautifully-designed, SEO-friendly, highly-performant WordPress sites for your clients.*

Earlier we mentioned the importance of establishing a starting template for all of your projects to improve your efficiency on repeat tasks.

Now a part of the WP Engine family, the Genesis Framework is our recommended starting point. Recently added features in Genesis 2.8 and 2.9 include theme configuration and onboarding tools, such as autoloading demo content.

The 2.10 release continues the refactoring efforts of 2.9, including moving settings and layout APIs to classes.

3.0 represents a significant release. In addition to further refactoring, we plan on adding in a few more capabilities to help with mobile including a special AMP integration the Genesis R&D team is working on in partnership with Google.


Alongside these changes to the Genesis Framework, we'll continue to release new StudioPress child themes that take advantage of the block editor, giving you the ability to produce beautifully-designed, SEO-friendly, highly-performant WordPress sites for your clients.

You can read more here about [getting started with WP Engine and the Genesis Framework](#).

### Genesis WP-CLI.

Genesis 2.10 includes [new WP-CLI commands](#) that help automate tasks such as checking for updates to the framework or completing a database upgrade after an update has occurred. DevKit's `wpe genesis` command gives you a convenient way to access those commands in your local development environment.


## Atomic Blocks.



### Advanced Columns Block

A powerful, flexible column system to build custom, full-page layouts for your posts and pages.


[View Block Demo](#)



### Newsletter Block

Add a customizable Mailchimp-powered email subscription form to your site and grow your audience.


[View Block Demo](#)



### Pricing Table Block

Flexible, customizable, easy-to-use pricing tables to showcase your products and prices.

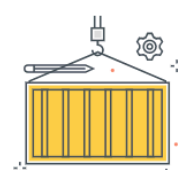
[View Block Demo](#)



### Post Grid Block

Add an eye-catching, full-width section with a big title, paragraph text, and a customizable button.


[View Block Demo](#)



### Container Block

Wrap several blocks into a section and add padding, margins, background colors and images.

[View Block Demo](#)



### Call-To-Action Block

Add an eye-catching, full-width section with a big title, paragraph text, and a customizable button.

[View Block Demo](#)

Also a part of the WP Engine family, Atomic Blocks builds on block-editor functionality and provides “ready made” blocks to help you create dynamic page layouts faster and with less code.

Tightly integrated with the Genesis Framework, Atomic Blocks remains under active development and will continue to introduce additional block types over time.





**Support &  
feedback.**



DevKit is still in beta status. If you run into questions or would like to provide feedback about your experience, there's a quick way to get in touch with your WP Engine DevKit team. Run the following command:

```
> wpe feedback
```

This launches a browser window with a feedback form you can use for direct contact.

## Going further.

Now that you have a better understanding of the steps and processes involved in a professional development workflow, it's time to start blending in the components that make the most sense for your business.

As you continue your learning journey, we recommend bookmarking our [Developer Guide](#), which includes links to in-depth resources for many of the topics mentioned in this document.



## About DevKit

The WP Engine Devkit is a combination of best-in-class development tools and processes geared towards providing a better overall developer experience and a more efficient workflow for creating powerful, great-looking digital experiences. DevKit includes a container-based local development environment, SSH Gateway access, seamless push & pull deployments, Genesis-specific functionality, and additional tools for building great WordPress projects faster. With all of these tools integrated together under a single roof, developers can focus their time and energy on coding, creating faster development cycles, and maximizing productivity, instead of searching for software.



## About WP Engine

WP Engine powers amazing digital experiences for websites and applications built on WordPress. The company's premium managed hosting platform provides the performance, reliability and security required by the biggest brands in the world, while remaining affordable and intuitive enough for smaller businesses and individuals. Companies of all sizes rely on WP Engine's award-winning customer service team to quickly solve technical problems and create a world-class customer experience. Founded in 2010, WP Engine is headquartered in Austin, Texas and has offices in Limerick, Ireland, San Francisco, California, San Antonio, Texas, and London, England. [wpengine.com](https://wpengine.com)

