Upgrade Guide

Migrating from Laravel Mix to Vite

> **Note**
> This upgrade guide does not cover all possible Mix use cases, such as Sass compilation. Please consult the Vite documentation for information on configuring Vite for these scenarios.

Update Laravel Framework

To make use of the new Vite integration, you will need to update to at least version `9.19.0` of the `laravel/framework`:

```
composer require laravel/framework:^9.19.0
```

Install Vite and the Laravel Plugin

First, you will need to install Vite and the Laravel Vite Plugin using your npm package manager of choice:

```
npm install --save-dev vite laravel-vite-plugin
```

You may also need to install additional Vite plugins for your project, such as the Vue or React plugins:

```
npm install --save-dev @vitejs/plugin-vue
```

```
npm install --save-dev @vitejs/plugin-react
```

Configure Vite

Create a `vite.config.js` file in the root of your project:

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';
// import react from '@vitejs/plugin-react';
// import vue from '@vitejs/plugin-vue';

export default defineConfig({
    plugins: [
        laravel([
            'resources/css/app.css',
            'resources/js/app.js',
        ]),
        // react(),
        // vue({
        //     template: {
        //         transformAssetUrls: {
        //             base: null,
        //             includeAbsolute: false,
        //         },
        //     },
        // }),
    ],
});
```

If you are building an SPA, you will get a better developer experience by removing the CSS entry point above and importing your CSS from Javascript.

Update Aliases

If you are migrating aliases from your `webpack.mix.js` file to your `vite.config.js` file, you should ensure that the paths start with `/`. For example, `resources/js` would become `/resources/js`:

```
export default defineConfig({
    plugins: [
        laravel([
            'resources/css/app.css',
            'resources/js/app.js',
        ]),
    ],
    resolve: {
        alias: {
            '@': '/resources/js'
        }
    }
});
```

For your convenience, the Laravel Vite plugin automatically adds an `@` alias for your `/resources/js` directory. If you do not need to customize your aliases, you may omit this section from your `vite.config.js` file.

Update NPM scripts

Update your NPM scripts in `package.json`:

```
  "scripts": {
-     "dev": "npm run development",
-     "development": "mix",
-     "watch": "mix watch",
-     "watch-poll": "mix watch -- --watch-options-poll=1000",
-     "hot": "mix watch --hot",
-     "prod": "npm run production",
-     "production": "mix --production"
+     "dev": "vite",
+     "build": "vite build"
  }
```

Vite compatible imports

Vite only supports ES modules, so if you are upgrading an existing application you will need to replace any `require()` statements with `import`. You may refer to this pull request for an example.

Inertia

Inertia makes use of a `require()` call that is more complex to replicate with Vite.

The following function can be used instead:

```
+ import { resolvePageComponent } from 'laravel-vite-plugin/inertia-helpers';

  createInertiaApp({
      title: (title) => `${title} - ${appName}`,
-     resolve: (name) => require(`./Pages/${name}.vue`),
+     resolve: (name) => resolvePageComponent(`./Pages/${name}.vue`, import.meta.glob('./Pages/**/*.vue')),
      setup({ el, app, props, plugin }) {
          return createApp({ render: () => h(app, props) })
              .use(plugin)
              .mixin({ methods: { route } })
              .mount(el);
      },
  });
```

Additionally, you should ensure you have updated to at least version `0.6.3` of the `inertia-laravel` package:

```
composer require inertiajs/inertia-laravel:^0.6.3
```

Update environment variables

You will need to update the environment variables that are explicitly exposed in your `.env` files and in hosting environments such as Forge to use the `VITE_` prefix instead of `MIX_`:

```
- MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
- MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
+ VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
+ VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

> **Note**
> You may optionally maintain the `MIX_` prefix by configuring Vite to use it.

You will also need to update these references in your JavaScript code to use the new variable name and Vite syntax:

```
-    key: process.env.MIX_PUSHER_APP_KEY,
-    cluster: process.env.MIX_PUSHER_APP_CLUSTER,
+    key: import.meta.env.VITE_PUSHER_APP_KEY,
+    cluster: import.meta.env.VITE_PUSHER_APP_CLUSTER,
```

Importing your CSS from your JavaScript entry point(s)

If you are building an SPA, you will get a better experience by importing your CSS from your JavaScript entry point(s), such as your `resources/js/app.js` entry point:

```
  import './bootstrap';
+ import '../css/app.css';
```

In development mode, Vite will automatically inject your CSS into the page. In production, a dedicated stylesheet will be generated that the `@vite` directive will load from the manifest.

Replace `mix()` with `@vite`

When using Vite, you will need to use the `@vite` Blade directive instead of the `mix()` helper.

This will automatically detect whether you are running in serve or build mode and include all of the required `<script>` and `<link rel="stylesheet">` for you:

```
- <link rel="stylesheet" href="{{ mix('css/app.css') }}">
- <script src="{{ mix('js/app.js') }}" defer></script>
+ @vite(['resources/css/app.css', 'resources/js/app.js'])
```

The entry points should match those used in your `vite.config.js`.

React

If you are using React and hot-module replacement, you will need to include an additional directive *before* the `@vite` directive:

```
@viteReactRefresh
@vite('resources/js/app.jsx')
```

This loads a React "refresh runtime" in development mode only, which is required for hot module replacement to work correctly.

JavaScript files containing JSX must use a `.jsx` extension

You will need to rename any `.js` files containing JSX to instead have a `.jsx` extension. If you need to rename your entry point then you should read the entry point docs to learn how to configure the Laravel plugin for your project.

See this tweet from Vite's creator for more information.

> **Note**
> If you are using Tailwind, remember to update the paths in your `tailwind.config.js` file.

Vue imports must include the `.vue` extension

```
- import Button from './Button';
+ import Button from './Button.vue';
```

Remove Laravel Mix

The Laravel Mix package can now be uninstalled:

```
npm remove laravel-mix
```

And you may remove your Mix configuration file:

```
rm webpack.mix.js
```

If you are using StyleCI and have ignored the `webpack.mix.js` file in your configuration, you may also wish to remove the ignore rule.

Update Test Helpers

If you are using the `$this->withoutMix();` helper in your tests, you should replace this with `$this->withoutVite()`:

```
- $this->withoutMix();
+ $this->withoutVite();
```

Vapor

If you are deploying your application to Laravel Vapor, there are a few things you will want to handle before deploying.

Ensure you have updated to at least version `1.40.0` of the Vapor CLI package:

```
composer require laravel/vapor-cli:^1.40.0
```

Next, if you are using the Vapor asset helper in your application, you only need to utilize the asset helper when you are referencing assets you don't want bundled, such as those that already live in your public directory.

If you want to use the asset helper with your Vite project, you will also need to ensure you have updated to the latest version:

```
npm install laravel-vapor@latest
```

Then you will need to specify the base URL for assets in your application's entry point, for example in your `resources/js/app.js,` like so:

```
- window.Vapor = require('laravel-vapor');
+ import Vapor from 'laravel-vapor';

+ window.Vapor = Vapor;
+ window.Vapor.withBaseAssetUrl(import.meta.env.VITE_VAPOR_ASSET_URL)
```

Optional: Configure Tailwind

If you are using Tailwind, perhaps with one of Laravel's starter kits, you will need to create a `postcss.config.js` file. Tailwind can generate this for you automatically:

```
npx tailwindcss init -p
```

Or, you can create it manually:

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
}
```

If you are using other PostCSS plugins, such as `postcss-import`, you will need to include them in your configuration.

Optional: Git ignore the build directory

Vite will place all of your build assets into a `build` subdirectory inside your public directory. If you prefer to build your assets on deploy instead of committing them to your repository, you may wish to add this directory to your `.gitignore` file:

```
/public/build
```

Optional: Update SSR configuration

You may remove your dedicated Laravel Mix SSR configuration:

```
rm webpack.ssr.mix.js
```

In most cases, you won't need a dedicated SSR configuration file when using Vite. You can specify your SSR entry point by passing a configuration option to the Laravel plugin:

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';

export default defineConfig({
    plugins: [
        laravel({
            input: 'resources/js/app.js',
            ssr: 'resources/js/ssr.js',
        }),
    ],
});
```

You may wish to add the following additional scripts to your `package.json`:

```
  "scripts": {
      "dev": "vite",
-     "build": "vite build"
+     "build": "vite build && vite build --ssr"
  }
```

If you prefer to build your assets on deploy instead of committing them to your repository, you may wish to add the SSR output directory to your `.gitignore` file:

```
/bootstrap/ssr
```

You may start the SSR server using `node`:

```
node bootstrap/ssr/ssr.mjs
```

Optional: Expose Vite port when using Laravel Sail

If you would like to run the `npm run dev` command in a Laravel Sail container, you will need to publish a port in your `docker-compose.yml` file:

```
ports:
    - '${APP_PORT:-80}:80'
+   - '${VITE_PORT:-5173}:${VITE_PORT:-5173}'
```

Wrapping up

You should now be able to build your assets using `dev` command. This will also invoke the Vite server and Vite will watch for file changes:

```
npm run dev
```

Alternatively, if you need to build files without watching or if you need to build them for production, you may use the `build` command:

```
npm run build
```

For further information on how to use Vite, please check out the Laravel Vite documentation.

Troubleshooting

If you have followed the upgrade guide, but are still having issues you should try the following steps:

- Run `php artisan view:clear` to clear any compiled view assets.
- If your development web server is running on HTTPS, check out the "Working With A Secure Development Server" section of the documentation.

Migrating from Vite to Laravel Mix

Install Laravel Mix

First, you will need to install Laravel Mix using your npm package manager of choice:

```
npm install --save-dev laravel-mix
```

Configure Mix

Create a `webpack.mix.js` file in the root of your project:

```
const mix = require('laravel-mix');

/*
 |--------------------------------------------------------------------------
 | Mix Asset Management
 |--------------------------------------------------------------------------
 |
 | Mix provides a clean, fluent API for defining some Webpack build steps
 | for your Laravel applications. By default, we are compiling the CSS
 | file for the application as well as bundling up all the JS files.
 |
 */

mix.js('resources/js/app.js', 'public/js')
    .postCss('resources/css/app.css', 'public/css', [
        //
    ]);
```

Update NPM scripts

Update your NPM scripts in `package.json`:

```
  "scripts": {
-     "dev": "vite",
-     "build": "vite build"
+     "dev": "npm run development",
+     "development": "mix",
+     "watch": "mix watch",
+     "watch-poll": "mix watch -- --watch-options-poll=1000",
+     "hot": "mix watch --hot",
+     "prod": "npm run production",
+     "production": "mix --production"
  }
```

Inertia

Vite requires a helper function to import page components which is not required with Laravel Mix. You can remove this as follows:

```
- import { resolvePageComponent } from 'laravel-vite-plugin/inertia-helpers';

  createInertiaApp({
      title: (title) => `${title} - ${appName}`,
-     resolve: (name) => resolvePageComponent(`./Pages/${name}.vue`, import.meta.glob('./Pages/**/*.vue')),
+     resolve: (name) => require(`./Pages/${name}.vue`),
      setup({ el, app, props, plugin }) {
          return createApp({ render: () => h(app, props) })
              .use(plugin)
              .mixin({ methods: { route } })
              .mount(el);
      },
  });
```

Update environment variables

You will need to update the environment variables that are explicitly exposed in your `.env` files and in hosting environments such as Forge to use the `MIX_` prefix instead of `VITE_`:

```
- VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
- VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
+ MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
+ MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

You will also need to update these references in your JavaScript code to use the new variable name and Node syntax:

```
-    key: import.meta.env.VITE_PUSHER_APP_KEY,
-    cluster: import.meta.env.VITE_PUSHER_APP_CLUSTER,
+    key: process.env.MIX_PUSHER_APP_KEY,
+    cluster: process.env.MIX_PUSHER_APP_CLUSTER,
```

Remove CSS imports from your JavaScript entry point(s)

If you are importing your CSS via JavaScript, you will need to remove these statements:

```
- import '../css/app.css';
```

Replace `@vite` with `mix()`

You will need to replace the `@vite` Blade directive with `<script>` and `<link rel="stylesheet">` tags and the `mix()` helper:

```
- @viteReactRefresh
- @vite('resources/js/app.js')
+ <link rel="stylesheet" href="{{ mix('css/app.css') }}">
+ <script src="{{ mix('js/app.js') }}" defer></script>
```

Remove Vite and the Laravel Plugin

Vite and the Laravel Plugin can now be uninstalled:

```
npm remove vite laravel-vite-plugin
```

Next, you may remove your Vite configuration file:

```
rm vite.config.js
```

You may also wish to remove any `.gitignore` paths you are no longer using:

```
- /bootstrap/ssr
- /public/build
```