Table of Contents

[VITE 2](#_Toc168914646)

[HOW VITE MAKES THE DEVELOPMENT EXPERIENCE FASTER? 2](#_Toc168914647)

[VITE VERSUS WEBPACK 2](#_Toc168914648)

[CREATE SIMPLE VITE PROJECT 2](#_Toc168914649)

[Scaffolding First Vite Project (USING REACT) 2](#_Toc168914650)

[PUBLISHING THE VITE APPLICATION 5](#_Toc168914651)

[SITE CONFIGURATION 7](#_Toc168914652)

[CREATING PLUG-IN VITE 8](#_Toc168914653)

# VITE

* Vite is a French word means “fast”. Vite was originally designed to improve the development experience of “Vue” js project
* Primarily Vite aims to provide a more lightweight and a faster development experience than other tools including webpack.

## HOW VITE MAKES THE DEVELOPMENT EXPERIENCE FASTER?

* The reason that makes it so fast for development is its use of native ES modules in the browser(Native ES or ECMAScript modules are a standardized format for organizing and sharing JavaScript code)

|  |  |
| --- | --- |
| * In modern browsers - modules are supported natively so we can load an module's JavaScript file directly in a script tag by adding the type to be module .   **<script src=”../main.tsx” type=”module”/>**   * This allows us to load that module JavaScript file without the need for a bundler or a transpiler and it's this modern browser feature that takes advantage of to provide us with such a fast development experience. * **The approach it takes is to use native ES modules like this during development, but then to bundle**   **the files into a production bundle when the application is ready to be deployed.**   * Ultimately, this approach allows to start up and rebuild projects much faster than other tools |  |

## VITE VERSUS WEBPACK

* Vite and Webpack both offer similar functionality.
* They're primarily bundlers to turn many separate JavaScript files into a smaller number of production assets, and they're both also minify and version the results.

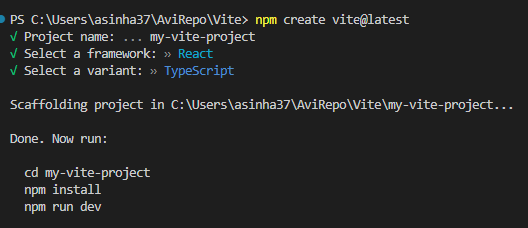
|  |  |  |
| --- | --- | --- |
| **Criteria** | **Vite** | **Webpack** |
| **Development** | Designed for modern, fast development | Suitable for small to large-scale projects |
| **Performance** | Utilizes native ES modules for faster bundling and loading | Slower bundling and loading times |
| **Hot Module Replacement** | Built-in support for fast module replacement | Requires additional configuration and plugins |
| **Configuration** | Minimal configuration setup required | Extensive configuration setup and management |
| **Dependency Resolution** | Uses built-in esbuild for faster dependency resolution | Requires loaders and plugins for various file types |
| **Ecosystem** | Limited plugin ecosystem compared to Webpack | Large and mature plugin ecosystem |
| **Code Splitting** | Automatic code splitting based on ES modules | Supports code splitting but requires manual configuration |
| **Production Build** | Generates optimized, lightweight bundles | Generates larger, more complex bundles |
| **Build Speed** | Extremely fast initial build time | Slower initial build time, especially for large projects |

## CREATE SIMPLE VITE PROJECT

* Need minimum Node version : 16

### Scaffolding First Vite Project (USING REACT)

|  |  |
| --- | --- |
| Command | npm create vite@latest |



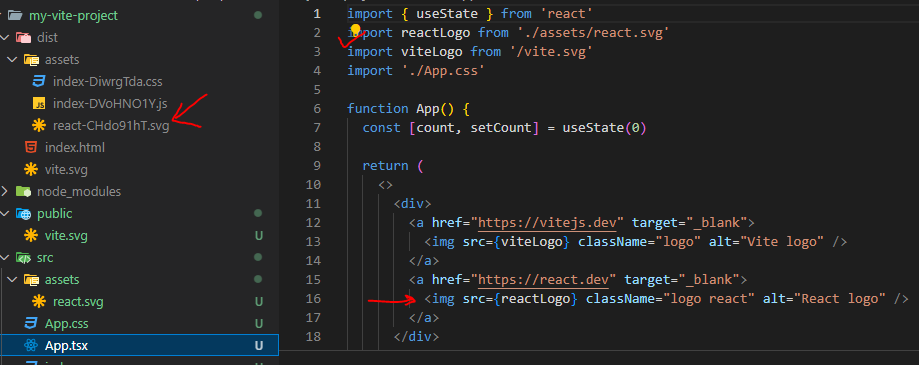
#### FILES IN A VITE PROJECT

|  |  |  |  |
| --- | --- | --- | --- |
|  | * **vite.config.ts** : This is used configure Vite * **tsconfig.json** : We can configure various TypeScript compiler options specific to Vite project. The include property specifies the files that should be included for compilation, while the exclude property specifies the files and folders that should be excluded.. * **tsconfig.node.json:** file needed in a Vite project, as the configuration is primarily focused on optimizing development and fast HMR (Hot Module Replacement) capabilities.   APPLICATION SOURCE FILES:   * By default, it creates us a single page application. We will have one (index.html) this is the only HTML file that we run.  |  | | --- | | <!doctype html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <link rel="icon" type="image/svg+xml" href="/vite.svg" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <title>Vite + React + TS</title>  </head>  <body>  <div id="root"></div>  **<script type="module" src="/src/main.tsx"></script>**  </body>  </html> |  * It has a script tag with the type of module, when we are in development mode, it uses JavaScript modules to be able to reload the files at development time in an incredibly fast way. * After we built the app with **npm run build** - will actually remove the script tag and reference all the code from a bundle - similar to Webpack.  |  | | --- | | <!doctype html>  <html lang="en">  <head>  <meta charset="UTF-8" />  <link rel="icon" type="image/svg+xml" href="/vite.svg" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <title>Vite + React + TS</title>  <script type="module" crossorigin src="/assets/index-DVoHNO1Y.js"></script>  <link rel="stylesheet" crossorigin href="/assets/index-DiwrgTda.css">  </head>  <body>  <div id="root"></div>  </body>  </html> |      * The philosophy behind vite is - we 'll be developing and testing the application with a modern browser. When it us published it will be supported by slightly older browsers. * Command : **npm run build** will create a production build which creates a “dist” folder * **Note : The dist folder is git ignored by default so it won't get checked in when we commit our code to gitand it's cleared and rewritten each time we do a build.** |

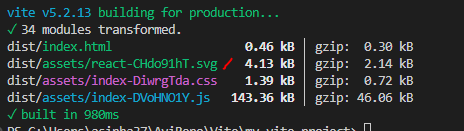
#### TREE SHAKING IN VITE

* In Vite, tree shaking refers to the process of eliminating unused code from our JavaScript or TypeScript modules during the bundling phase. It helps reduce the size of the final bundle by removing any dead code that is not being used in your application.
* Vite leverages the native ES module (ESM) syntax, which inherently supports tree shaking. When we import modules using the import statement, Vite can analyze the dependencies and eliminate any unused code.

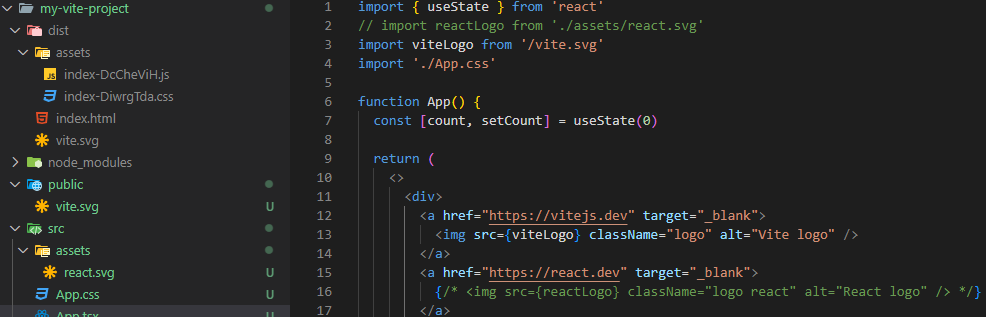
EXAMPLE



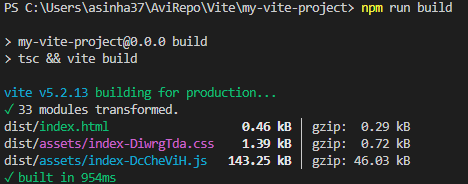
BUILD



CASE2 : When reactLogo import is removed

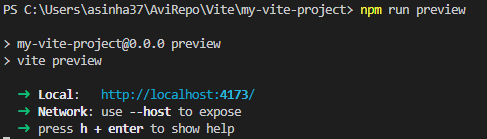


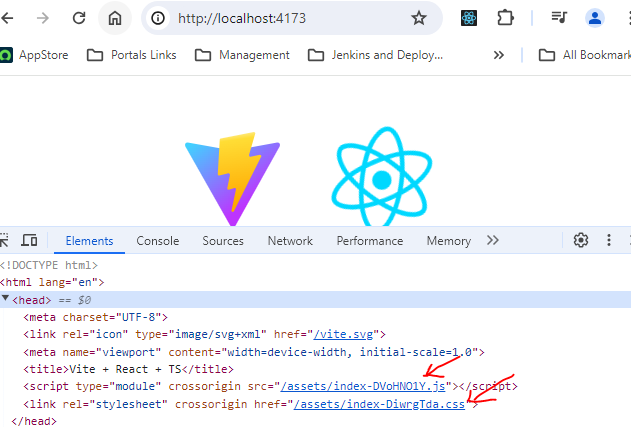
BUILD



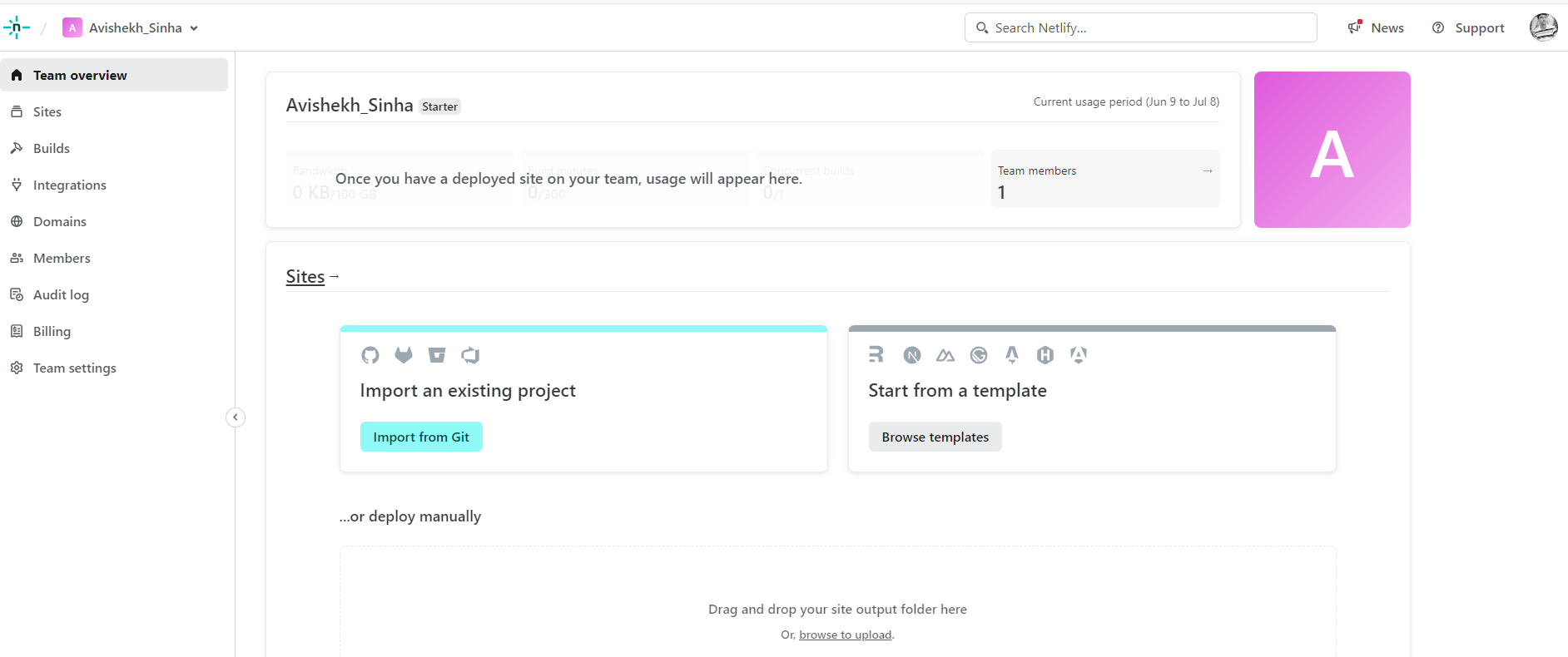
## PUBLISHING THE VITE APPLICATION

* Netlify is a cloud-based platform that provides a comprehensive solution for web development, hosting, and deployment. It offers features and services aimed at simplifying the process of building, deploying, and managing websites or web applications.
* Step 1: For production build run command : **npm run build** . This will compile the TS 🡪 JS
* Step 2: To validate the production build: **npm run preview**

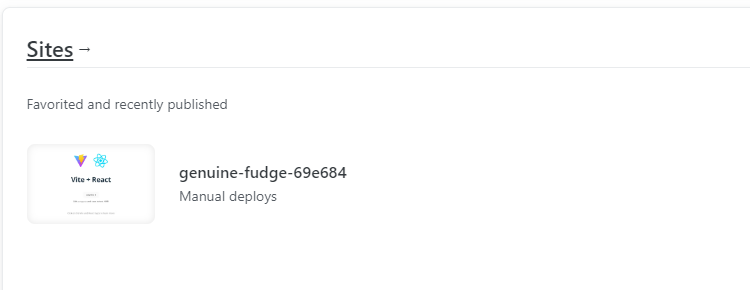




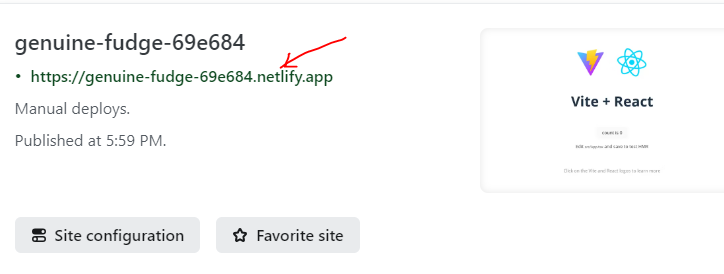
* Step 3: Login to <https://app.netlify.com/>
* Step 4: Navigate to “Team Overview” Page 🡪 Sites



Step 5: Upload the “dist” in upload file area



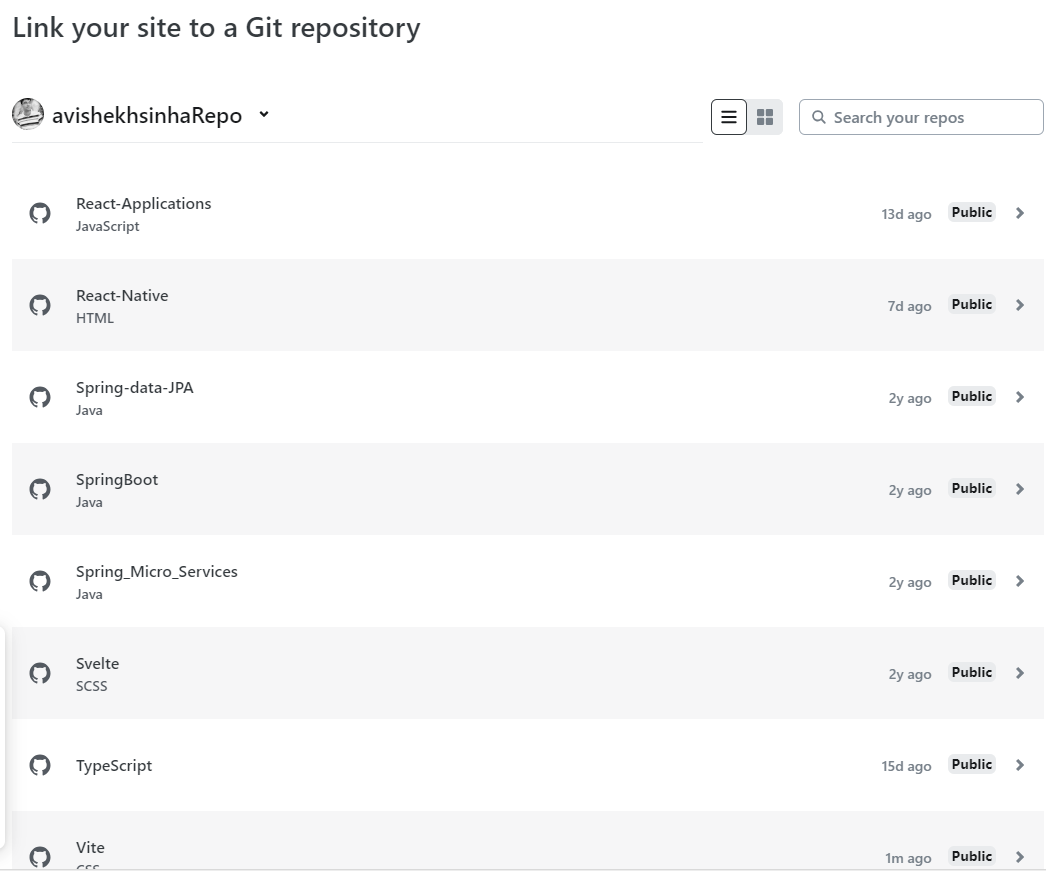
Step 6: Navigate to the site and access it by link assigned to the app



### SITE CONFIGURATION

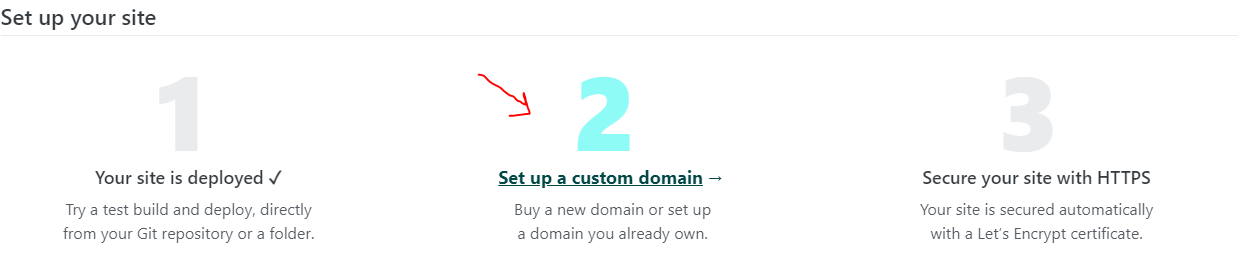
#### SETTING UP CI/CD ON NETLIFY

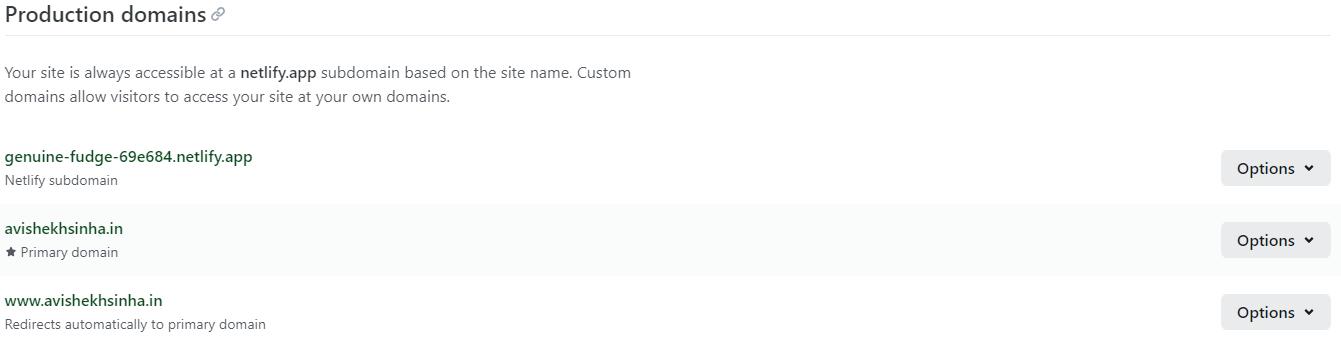
* Step 1 : Navigate to Site 🡪 Site Configuration
* Step 2: Go to Build & Deploy 🡪 Link it to repository.
* Step 3: Once The Repo has been configured we can able to view the Repos



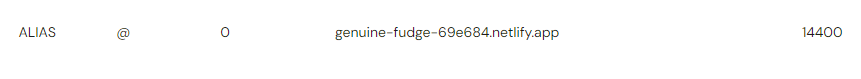
Step 4: Lets configure the CI/ CD for the Vite Repo

#### CONFIGURE THE CUSTOM DOMAIN





**CONFIGURATION IN DOMAIN PROVIDER**





## CREATING PLUG-IN VITE