# Kit Utilities

👉 Read more in Docs > Guide > Going Further > Kit.

## Utilities

### Modules

source code

- `installModule(module, inlineOptions)`

### Programmatic Usage

source code

- `loadNuxt(loadOptions)`
- `buildNuxt(nuxt)`
- `loadNuxtConfig(loadOptions)`

### Compatibility

source code

- `checkNuxtCompatibility(constraints)`
- `assertNuxtCompatibility(constraints)`
- `hasNuxtCompatibility(constraints)`
- `isNuxt2()`
- `isNuxt3()`
- `getNuxtVersion()`

## Auto-imports

source code

- `addImports(imports)`
- `addImportsDir(importDirs, { prepend? })`
- `addImportsSources(importSources)`

## Components

source code

- `addComponentsDir(dir)`
- `addComponent(componentObject)`

## Context

source code

- `useNuxt()`
- `tryUseNuxt()`

## Pages

source code

- `extendPages (callback: pages => void)`
- `extendRouteRules (route: string, rule: NitroRouteConfig, options: ExtendRouteRulesOptions)`
- `addRouteMiddleware (input: NuxtMiddleware | NuxtMiddleware[], options: AddRouteMiddlewareOptions)`

## Plugins

source code

- `addPlugin(pluginOptions, { append? })`
- `addPluginTemplate(pluginOptions, { append? })`

## Templates

source code

- `addTemplate(templateOptions)`
- `addTypeTemplate(templateOptions)`
- `updateTemplates({ filter?: ResolvedNuxtTemplate => boolean })`

## Nitro

source code

- `addServerHandler (handler)`
- `addDevServerHandler (handler)`
- `useNitro()` (only usable after `ready` hook)
- `addServerPlugin`
- `addPrerenderRoutes`

## Resolving

source code

- `resolvePath (path, resolveOptions?)`
- `resolveAlias (path, aliases?)`
- `findPath (paths, resolveOptions?)`
- `createResolver (base)`

## Logging

source code

- `useLogger(scope?)`

## Builder

source code

- `extendWebpackConfig(callback, options?)`
- `extendViteConfig(callback, options?)`
- `addWebpackPlugin(webpackPlugin, options?)`
- `addVitePlugin(vitePlugin, options?)`

# Examples

## Accessing Nuxt Vite Config

If you are building an integration that needs access to the runtime Vite or webpack config that Nuxt uses, it is possible to extract this using Kit utilities.

Some examples of projects doing this already:

- histoire
- nuxt-vitest
- @storybook-vue/nuxt

Here is a brief example of how you might access the Vite config from a project; you could implement a similar approach to get the webpack configuration.

```
import { loadNuxt, buildNuxt } from '@nuxt/kit'

// https://github.com/nuxt/nuxt/issues/14534
async function getViteConfig() {
  const nuxt = await loadNuxt({ cwd: process.cwd(), dev: false, overrides: { ssr: false } })
  return new Promise((resolve, reject) => {
    nuxt.hook('vite:extendConfig', (config, { isClient }) => {
      if (isClient) {
        resolve(config)
        throw new Error('_stop_')
      }
    })
    buildNuxt(nuxt).catch((err) => {
      if (!err.toString().includes('_stop_')) {
        reject(err)
      }
    })
  }).finally(() => nuxt.close())
}

const viteConfig = await getViteConfig()
console.log(viteConfig)
```

Edit on Github