Configuration

nuxt.config

The starting point for your Nuxt app remains your `nuxt.config` file.

Nuxt configuration will be loaded using `unjs/jiti` and `unjs/c12`.

Migration

1. You should migrate to the new `defineNuxtConfig` function that provides a typed configuration schema.

   Nuxt 2 **Nuxt 3**

   Nuxt 2

   ```
   export default {
     // ...
   }
   ```

2. If you were using `router.extendRoutes` you can migrate to the new `pages:extend` hook:

   Nuxt 2 **Nuxt 3**

Nuxt 2

```
export default {
  router: {
    extendRoutes (routes) {
      //
    }
  }
}
```

ESM Syntax

Nuxt 3 is an ESM native framework. Although `unjs/jiti` provides semi compatibility when loading `nuxt.config` file, avoid any usage of `require` and `module.exports` in this file.

1. Change `module.exports` to `export default`
2. Change `const lib = require('lib')` to `import lib from 'lib'`

Async Configuration

In order to make Nuxt loading behavior more predictable, async config syntax is deprecated. Consider using Nuxt hooks for async operations.

Dotenv

Nuxt has built-in support for loading `.env` files. Avoid directly importing it from `nuxt.config`.

Modules

Nuxt and Nuxt Modules are now build-time-only.

Migration

1. Move all your `buildModules` into `modules`.

2. Check for Nuxt 3 compatibility of modules.

3. If you have any local modules pointing to a directory you should update this to point to the entry file:

```
  export default defineNuxtConfig({
    modules: [
-     '~/modules/my-module'
+     '~/modules/my-module/index'
    ]
  })
```

If you are a module author, you can check out more information about module compatibility and our module author guide.

Directory Changes

The `static/` directory (for storing static assets) has been renamed to `public/`. You can either rename your `static` directory to `public`, or keep the name by setting `dir.public` in your `nuxt.config`.

👉 Read more in Docs > Guide > Directory Structure > Public.

TypeScript

It will be much easier to migrate your application if you use Nuxt's TypeScript integration. This does not mean you need to write your application in TypeScript, just that Nuxt will provide automatic type hints for your editor.

You can read more about Nuxt's TypeScript support in the docs.

📦 Nuxt can type-check your app using `vue-tsc` with `nuxi typecheck` command.

Migration

1. Create a `tsconfig.json` with the following content:

```
{
  "extends": "./.nuxt/tsconfig.json"
}
```

2. Run `npx nuxi prepare` to generate `.nuxt/tsconfig.json`.

3. Install Volar following the instructions in the docs.

Vue Changes

There are a number of changes to what is recommended Vue best practice, as well as a number of breaking changes between Vue 2 and 3.

It is recommended to read the Vue 3 migration guide and in particular the breaking changes list.

It is not currently possible to use the Vue 3 migration build with Nuxt 3.

Vuex

Nuxt no longer provides a Vuex integration. Instead, the official Vue recommendation is to use `pinia`, which has built-in Nuxt support via a Nuxt module. Find out more about pinia here.

A simple way to provide global state management with pinia would be:

Install the @pinia/nuxt module:

```
> yarn add pinia @pinia/nuxt
```

Enable the module in your nuxt configuration:

nuxt.config.ts

```
import { defineNuxtConfig } from 'nuxt/config';

export default defineNuxtConfig({
  modules: ['@pinia/nuxt']
})
```

Create a `store` folder at the root of your application:

store/index.ts

```ts
import { defineStore } from 'pinia'

export const useMainStore = defineStore('main', {
  state: () => ({
    counter: 0,
  }),
  actions: {
    increment() {
      // `this` is the store instance
      this.counter++
    },
  },
})
```

Create a plugin file to globalize your store:

plugins/pinia.ts

```ts
import { useMainStore } from '~/store'

export default defineNuxtPlugin(({ $pinia }) => {
  return {
    provide: {
      store: useMainStore($pinia)
    }
  }
})
```

If you want to keep using Vuex, you can manually migrate to Vuex 4 following these steps.

Once it's done you will need to add the following plugin to your Nuxt app:

plugins/vuex.ts

```ts
import store from '~/store'

export default defineNuxtPlugin(nuxtApp => {
  nuxtApp.vueApp.use(store);
})
```