**To Speed Up Linting And Validation During A Next.Js Build**

To speed up linting and validation during a Next.js build, you can implement the following strategies:

**1. Disable Linting During Build**

By default, Next.js runs ESLint during the build process. You can disable this step and run linting separately to save time:

*Copy the code:* // next.config.js

module.exports = {

eslint: {

ignoreDuringBuilds: true,

},

};

This skips linting during the build, allowing you to handle it manually or in a CI pipeline.

**2. Use Incremental Linting**

Tools like eslint --cache can significantly speed up linting by only checking files that have changed:

*Copy the code:* eslint . --cache

Add this to your scripts in package.json for faster linting:

Copy the code: "scripts": {

"lint": "eslint . --cache"

}

**3. Optimize ESLint Configuration**

* **Reduce Plugins/Rules**: Only include necessary plugins and rules in your .eslintrc file.
* **Ignore Unnecessary Files**: Use .eslintignore to exclude files or directories that don’t need linting (e.g., node\_modules, build output).
* *Example* .eslintignore:
* *Copy the code:* node\_modules/
* .next/
* dist/

**4. Parallelize Linting**

Use tools like [eslint-parallel](https://www.npmjs.com/package/eslint-parallel) to run linting tasks in parallel, leveraging multiple CPU cores:

*Copy the code:* npx eslint-parallel .

**5. Upgrade Dependencies**

Ensure you’re using the latest versions of Next.js, ESLint, and related plugins, as newer versions often include performance improvements.

By applying these strategies, you can significantly reduce the time spent on linting and validation during your Next.js build process.