

Continuous Integration (CI) with ESLint & Prettier

1. What is CI?

- **Definition:** Automating code integration → build, test, lint on every push.
- **Goal:** Ensure main branch is always stable & deployable.

Flow:

```
Push code → GitHub Actions → Install deps → Lint/Build → Feedback
```

2. Why ESLint & Prettier?

- **ESLint:** Catches syntax errors & enforces rules.
- **Prettier:** Formats code consistently.
- **Together:** Maintain clean, readable, error-free codebase.

3. Setup in MERN Project

(A) Client (React + Vite)

- Install:

```
cd client
npm install eslint prettier eslint-plugin-react-hooks eslint-plugin-react-refresh --save-dev
```

- Configs: `eslint.config.js`

```
import js from '@eslint/js';
import globals from 'globals';
import reactHooks from 'eslint-plugin-react-hooks';
import reactRefresh from 'eslint-plugin-react-refresh';

export default [
  { ignores: ['dist'] },
  {
    files: ['**/*.js', '**/*.jsx'],
  },
];
```

```

languageOptions: {
  ecmaVersion: 'latest',
  sourceType: 'module',
  globals: globals.browser,
  parserOptions: { ecmaFeatures: { jsx: true } },
},
plugins: { 'react-hooks': reactHooks, 'react-refresh': reactRefresh },
rules: {
  ...js.configs.recommended.rules,
  ...reactHooks.configs.recommended.rules,
  'no-unused-vars': ['warn', { varsIgnorePattern: '^[_A-Z_]' }],
  'react-refresh/only-export-components': ['warn', {
allowConstantExport: true }]],
  },
},
];

```

`.prettierrc`

```
{ "semi": true, "singleQuote": true, "printWidth": 100 }
```

(B) Server (Node + Express)

- Install:

```

cd server
npm install eslint prettier --save-dev

```

- Configs: `eslint.config.js`

```

export default [
  {
    files: ['**/*.js'],
    rules: { semi: 'error', 'no-unused-vars': 'warn' },
  },
];

```

`.prettierrc` → same as client.

4. Package Scripts

Add in **both client & server**:

```
"scripts": {
  "lint": "eslint . --ext .js,.jsx",
  "lint:fix": "eslint . --ext .js,.jsx --fix",
  "format": "prettier --write ."
}
```

5. GitHub Actions Workflow

`.github/workflows/ci.yml`

```
name: build-and-lint

on:
  push:
    branches: [main, dev]
  pull_request:
    branches: [main, dev]

jobs:
  build-and-lint:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repo
        uses: actions/checkout@v3

      - name: Setup Node
        uses: actions/setup-node@v3
        with:
          node-version: 20

      - name: Install backend deps
        working-directory: server
        run: npm ci

      - name: Lint backend
        working-directory: server
        run: npm run lint -- --max-warnings=0

      - name: Install frontend deps
        working-directory: client
        run: npm ci

      - name: Lint frontend
        working-directory: client
```


```
run: npm run lint -- --max-warnings=0

- name: Build backend Docker image
  run: docker build -t backend ./server

- name: Build frontend Docker image
  run: docker build -t frontend ./client
```

6. Applying in MERN Workflow

- Push code → CI runs automatically.
- ESLint catches errors before merging.
- Prettier enforces consistent style.
- Docker builds confirm images are valid.

 **Summary:** CI with GitHub Actions ensures every commit is linted, formatted, and build-checked before integration, keeping the MERN codebase reliable.