



Most Widely Used Conventional Commit Tags

Tag	Purpose / When to Use	Example Commit Message
<code>init</code>	Initial project setup	<code>init: setup Next.js project with Tailwind CSS</code>
<code>setup</code>	Folder structure or environment setup	<code>setup: add folder structure and environment setup</code>
<code>feat</code>	Adding a new feature or functionality	<code>feat: implement navbar and routing</code>
<code>feat(api)</code>	Adding or integrating an API-related feature	<code>feat(api): add user card component with API integration</code>
<code>fix</code>	Fixing a bug or error	<code>fix: resolve API fetch error on 404 status</code>
<code>refactor</code>	Code improvement without changing behavior	<code>refactor: extract reusable button component</code>
<code>style</code>	Formatting or visual changes without logic change	<code>style: adjust padding and colors in header</code>
<code>ui</code>	Improving user interface or layout	<code>ui: improve responsive layout for mobile screens</code>
<code>perf</code>	Optimizing performance	<code>perf: optimize re-renders using React.memo</code>
<code>docs</code>	Documentation or README updates	<code>docs: add project overview and screenshots in README</code>
<code>test</code>	Adding or updating tests	<code>test(user): add unit tests for OTP verification</code>

Tag	Purpose / When to Use	Example Commit Message
<code>chore</code>	Routine maintenance (config, lint, etc.)	<code>chore(env): add JWT secret and expiration to env file</code>
<code>ci</code>	Continuous Integration / Deployment setup	<code>ci: configure GitHub Actions for deployment</code>
<code>build</code>	Build system or dependency changes	<code>build: update dependencies and fix build warnings</code>
<code>deploy</code>	Deployment-related changes	<code>deploy: deploy project on Vercel</code>
<code>revert</code>	Reverting a previous commit	<code>revert: undo navbar state management change</code>
<code>security</code>	Patching or improving security	<code>security: sanitize user input in login form</code>
<code>config</code>	Modifying configuration files	<code>config: update ESLint and Prettier rules</code>



Example Professional Commits:

- `feat(auth): implement OTP verification and token generation`
- `fix(login): correct OTP expiry logic`
- `docs(readme): update API usage instructions`
- `chore(env): add JWT secret and expiration to env file`
- `ci: configure GitHub Actions for deployment`
- `style(auth): format login controller for readability`
- `test(user): add unit tests for OTP verification`

Best Practices to Stand Out

- Use **present tense**: `add` not `added`, `fix` not `fixed`.
- Keep the **subject line under 72 characters**.

- Add **scope** in parentheses if possible: `feat(auth): ...`, `fix(api): ...`.