Conventional Commits Guide

Ali Emre Pamuk

April 2025

Contents

1 Introduction

Conventional Commits is a lightweight convention for writing consistent and meaningful commit messages. It helps automate release notes, semantic versioning, and project maintenance while improving readability and collaboration.

2 Commit Message Format

```
<type>[optional scope]: <short description>

[optional longer body]

[optional footer(s)]
```

Example

```
feat(auth): enable JWT tokens

Added JWT generation on successful login to replace sessions.

BREAKING CHANGE: session management is removed
Closes #42
```

3 Types

- **feat** – Adds a new feature `feat: add search bar`
- **fix** – Fixes a bug `fix: resolve login crash`
- **docs** – Documentation only changes `docs: update README`
- **style** – Formatting and non-functional code changes `style: format files with Prettier`
- **refactor** – Code refactoring with no behavior change `refactor: simplify user validation`
- **test** – Adds or updates tests `test: add auth tests`
- **chore** – Maintenance tasks like dependency updates `chore: update eslint config`

4 Scopes

Scopes define what part of the code the commit affects. They're optional but recommended.

```
feat(api): support user roles
fix(db): escape query inputs
```

5 Body and Footer

The body offers context and reasoning behind the change. Footers can include metadata:

- `BREAKING CHANGE:` for incompatible changes
- `Closes #123:` to auto-close issues or PRs

```
BREAKING CHANGE: API returns 401 instead of 403
Closes #123
```

6 Why Use Conventional Commits?

- **Clarity** – Understand what each commit does at a glance
- **Automation** – Enables changelog generation and semantic versioning
- **Consistency** – Helps teams write uniform commit messages
- **Integration** – Compatible with tools like Commitizen, semantic-release, etc.

Conclusion

Using Conventional Commits promotes better collaboration, easier automation, and cleaner project history. By following a predictable format, your commit messages can serve as documentation, changelog, and release notes all in one.