



PATCH

FIX

FEAT

BREAKING CHANGE



MINOR



MAJOR

FIX → **PATCH**

FEAT → **MINOR**

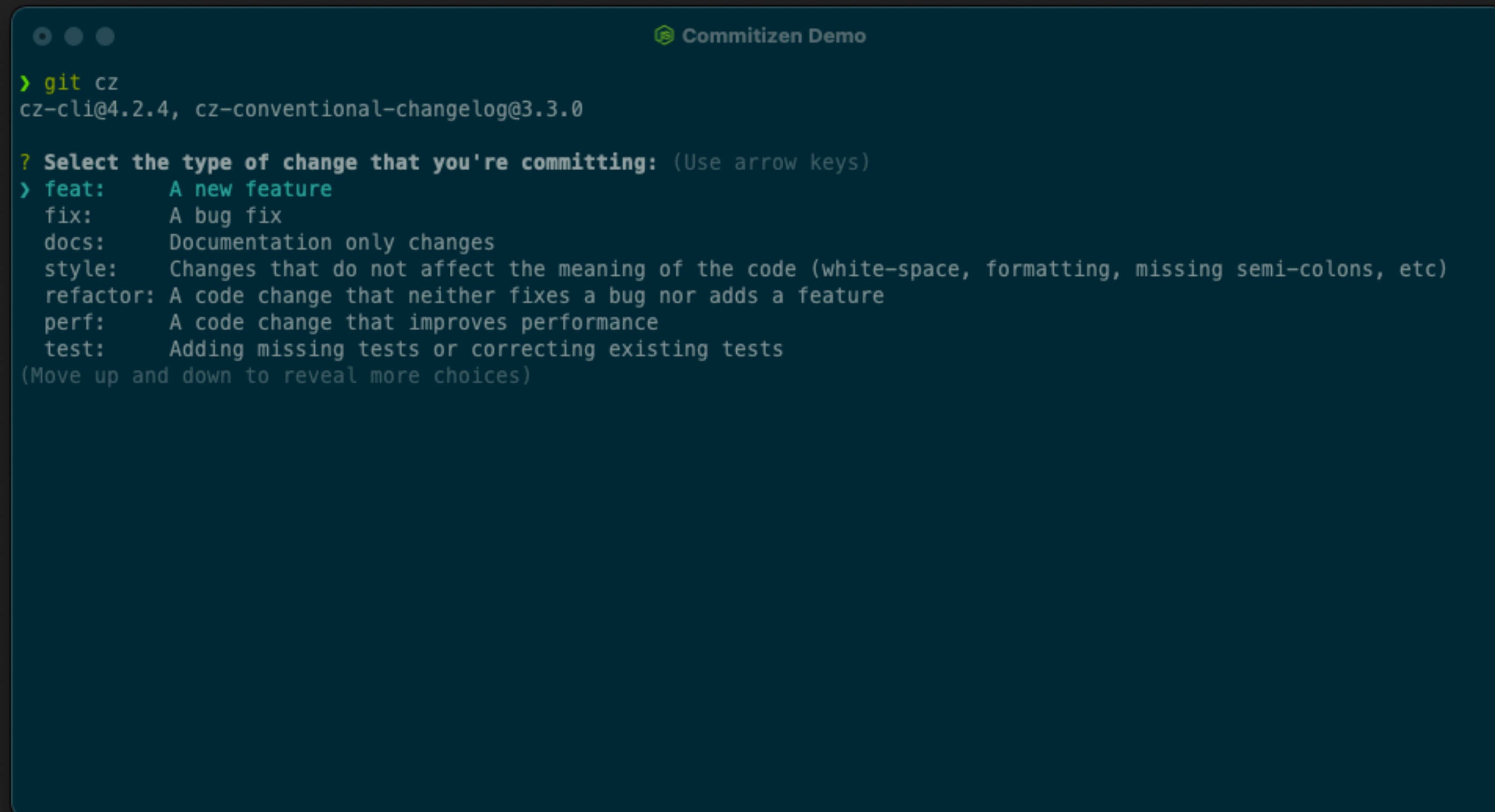
BREAKING CHANGE → **MAJOR**

TOOLS TO HELP EASE ADOPTION OF CONVENTIONAL COMMITS

Commitizen - CLI to help format commits

Commitlint - Lint your commits

Husky - Git Hooks

A terminal window titled "Commitizen Demo" showing the execution of the 'git cz' command. The terminal displays the version information 'cz-cli@4.2.4, cz-conventional-changelog@3.3.0' and a prompt to select a commit type. The available options are listed: 'feat: A new feature', 'fix: A bug fix', 'docs: Documentation only changes', 'style: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)', 'refactor: A code change that neither fixes a bug nor adds a feature', 'perf: A code change that improves performance', and 'test: Adding missing tests or correcting existing tests'. A note at the bottom indicates that arrow keys can be used to navigate through the choices.

```
> git cz
cz-cli@4.2.4, cz-conventional-changelog@3.3.0

? Select the type of change that you're committing: (Use arrow keys)
> feat:      A new feature
  fix:       A bug fix
  docs:      Documentation only changes
  style:     Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)
  refactor:  A code change that neither fixes a bug nor adds a feature
  perf:      A code change that improves performance
  test:      Adding missing tests or correcting existing tests
(Move up and down to reveal more choices)
```