Git commit messaging

How to write a Git commit message:

1. Begin the commit message with a short title that summarizes the change.
2. Start the title with an imperative present active verb: `Add`, `Drop`, `Fix`, `Refactor`, `Optimize`, etc.
3. Format the title: start with a capital word, use up to `50 characters`, and end without a period.
4. Optionally follow the title by a blank line, then a more thorough description.

Title examples

Title examples of good commit messages:

- Add feature for ...
- Drop feature for ...
- Fix bug when ... is missing
- Start feature flag
- Stop feature flag
- Refactor ... for clarity
- Optimize ... for speed and memory

Keyword Strategy

We will use these keywords because they use the following:

- imperative moods
- present tense
- an active voice
- verbs

Keywords

- `Add` = Create a capability e.g. feature, test, dependency.
- `Drop` = Delete a capability e.g. feature, test, dependency.
- `Fix` = Fix an issue e.g. bug, typo, accident, misstatement.
- `Bump` = Increase the version of something e.g. a dependency.
- `Make` = Change the build process, or tools, or infrastructure.

- `Start` = Begin doing something; e.g. enable a toggle, feature flag, etc.
- `Stop` = End doing something; e.g. disable a toggle, feature flag, etc.
- `Refactor` = A change that MUST be just refactoring.
- `Reformat` = A change that MUST be just formatting, e.g. omit whitespace.
- `Rephrase` = A change that MUST be just textual, e.g. edit a comment, doc, etc.
- `Optimize` = A change that MUST be just about performance, e.g. speed up code.
- `Document` = A change that MUST be only in the documentation, e.g. help files.

Semantic versioning

We use semantic versioning for many of our projects:

- `Add` = Increment *SemVer* MINOR version.
- `Drop` = Increment *SemVer* MAJOR version.
- `Fix`, `Refactor`, `Reformat`, `Rephrase`, `Optimize`, etc. = Increment *SemVer* PATCH version.

Rules

Capitalize the title.

- Right: Add feature
- ~~Wrong: add feature~~

Do not end the title with a period:

- Right: Add feature
- ~~Wrong: Add feature.~~

Use imperative mood: present tense, active voice, and lead verb.

- Right: Add feature
- ~~Wrong: Adds feature (this is indicative mood, not imperative mood)~~
- ~~Wrong: Added feature (this is past tense, not present tense)~~
- ~~Wrong: Adding feature (this lead is a gerund, not a verb)~~
- ~~Wrong: Feature added (this is passive voice, not active voice)~~

It is best practice to keep the title within `50 characters`.

It is best practice use a blank line after the title.

It is best practice wrap the body at 72 characters. This is the same convention as writing an

email message.

For more about these see How to Write a Git Commit Message

Purpose and Reasoning

We primarily care that our team communicates effectively with an aligned understanding. The verbs above are helpful because they're they're easy to read, easy to type, align with best practice and are clear in many cultures.

Will will reject the following formats:

We will reject some kinds of git commit message formats using our linting commit hook

- [bug] ...
- (release) ...
- #12345 ...
- jira:// ...
- docs: ...

We reject the commit style of projets such as Angular, Commitizen defaults, etc.

- Because these use a leading tag that is sometimes a word, sometimes an abbreviation, sometimes a plural noun, etc.
- Examples are using "feat" for feature, "docs" for document, "perf" for performance improvement, etc.
- Instead we use "Add" for adding a feature, "Document" for documenting help, "Optimize" for performance improvement, etc.
- Active verbs are easier to skim, and easier to use for people from other cultures who may be less-comfortable using English.

We reject using an id number or URL in the title.

- We use fully-qualified URLs in the commit message body.
- This is because many of our projects use multiple tracking systems, and multiple ways of launching a URL.
- We want URL tracking to be easy to use by a wide range of systems, scripts, and teams.

Use task tracking links

We will connect a git commit to a task tracking system or web page that explains more.

For example, we use Pivotal Tracker, Invision and Confluence. To keep track of these, we will write git commit body messages that lists each URL, one per line because this is easy to parse.

Example:

```
Add feature foo

See: [Request for help with sign in](https://github.com/user/repo/issues/789)
See: [Add feature ...](https://jira.com/tasks/123)
See: [Invision ....](https://wikipedia/quicksort)
```