# Contribution Guidelines for Fluent-Bit

We build Open Source software and we invite everyone to join us and contribute. So if you are interested into participate, please refer to the guidelines below.

## Developer Guide

[Developer Guide with code examples](DEVELOPER\_GUIDE.md).

## GIT Repositories

All code changes and submissions happens on [Github](http://github.com), that means that to start contributing you should clone the target repository, perform local changes and then do a Pull Request. For more details about the workflow we suggest you check the following documents:

- https://help.github.com/articles/using-pull-requests

- https://help.github.com/articles/creating-a-pull-request

## Coding Style

Our development coding style for C is based on the Apache C style guidelines, we use similar rules, to get more details about it please check the following URL:

- https://httpd.apache.org/dev/styleguide.html

You have to pay attention to the code indentation, tabs are 4 spaces, spaces on conditionals, etc. If your code submission is not aligned, it will be rejected.

### General requirements

#### Line Length

Fluent Bit source code lines length should not exceed 90 characters.

#### Braces usage on conditionals, loops and functions:

Always append braces to a conditional or loop expression, e.g:

```c

if (ret == -1) {

return -1;

}

```

no matters if the code under the conditional is just one line, we need braces. Note that the opening brace is on the right side of the conditional and \_\_not\_\_ in the next line. Same rule applies for \_while\_() and do while() loop iterators

For \_\_if\_\_ and \_\_else\_\_ always respect a new line after the opening brace:

```c

if (ret == -1) {

return -1;

}

else if (ret == 0) {

return 0;

}

```

For \*\*function definitions\*\* the brace position is different, the opening brace is \_\_always\_\_ in the next line, e.g:

```c

int flb\_something(int a, int b)

{

return a + b;

}

```

### Variable definitions

Variables must be declared at the beginning of a function and not in the middle of the code, the following example demonstrate the wrong way to do it:

```c

int flb\_something(int a, int b)

{

if (a > 10) {

return 1;

}

else {

int ret;

ret = a + b;

return ret;

}

}

```

the proper way is to perform the variable definitions on top:

```c

int flb\_something(int a, int b)

{

int ret;

if (a > 10) {

return 1;

}

else {

ret = a + b;

return ret;

}

}

```

### Functions and nested levels

If your function is too long where many nested levels exists, consider to split your function in different ones and declare the spitted parts as static functions if they don't be intended to be called out of the scope of the source code file in question.

### Comments in the code

Commenting code is always encouraged, that makes things easier to the reader to understand what the code is doing or aims to do.

In Fluent Bit, every code comment starts with a slash asterisk ```/\*``` and ends with a asterisk slash ```\*/```. If the text in the comment is longer than 80 characters, append a new commented line. We use the following format depending on the case:

#### Single line comment

```C

/\* This is my comment \*/

```

#### Multiline comment

```c

/\*

\* This is my comment which is longer than 80 characters, so we must use the

\* multi-line type comments.

\*/

```

## Commit Changes

When you commit your local changes in your repository (before to push to Github), we need you take care of the following:

- Your principal commit message (one line subject) \*\*must be\*\* prefixed with the core section name in lowercase plus a colon. If you are fixing an call from the engine the commit message should be:

```

engine: fix handling of abc

```

Expanding a bit the example feature message we could use the following command:

> $ git commit -a -s

>

> engine: fix handling of abc

>

> This patch fix a problem when managing the flush buffer of ABC output plugin. It adds

> a new routines to check proper return values and validate certain exceptions.

>

> the patch have been tested using tools A & B.

>

> Signed-off-by: Your Name <your@email.com>

If you want to see a real example, run the following command:

> $ git log 54ea8d0b164d949745b5f4b83959400469737b45

Your patches should be fully documented. That will make the review process faster for us and a faster merge for you.

Common components prefix are:

- utils:

- pack:

- sds:

- http\_client:

As you can see prefixes are basically the file name of the source code file under [src](https://github.com/fluent/fluent-bit/tree/master/src) directory without the file prefix <u>flb\_</u>.

When committing changes to code that's related to some plugins, the commit subject must be prefixed with the name of the plugin being changed, e.g:

- in\_stdin:

- out\_http:

- out\_kafka:

please refer to the [plugins](https://github.com/fluent/fluent-bit/tree/master/plugins) directory as a reference

- One single commit \*\*must not\*\* include changes to files that are different from the component specified in the subject, e.g: If you are extending flb\_utils.c file, the git patch should not touch any other file than flb\_utils.c or flb\_utils.h.

- One single commit \*\*must not\*\* include multiple prefixes to specify different areas being touched.

- The subject of the commit \*\*must not\*\* be longer than 80 characters.

- On the commit body, each line \*\*must not\*\* be longer than 80 characters.

- On most of cases we want full description about what your patch is doing, the patch description should be self descriptive.. like for dummies. Do not assume everybody knows what you are doing and on each line do not exceed 80 characters.

- When running the \_\_git commit\_\_ command, make sure you are using the \_\_-s\_\_ flag, that will add a Signed-off comment in the patch description. If your commit is not signed-off, Github DCO check will fail and your contribution will not be reviewed until that get's fixed.

## Licensing

[Fluent-Bit](http://fluentbit.io) is an Open Source project and all it code base \_must\_ be under the terms of the [Apache License v2.0](http://www.apache.org/licenses/LICENSE-2.0). When submitting changes to the core or any new plugin, you agreed to share that code under the license mentioned. All your source code files must have the following header:

```

/\* Fluent Bit

\* ==========

\* Copyright (C) 2019-2020 The Fluent Bit Authors

\* Copyright (C) 2015-2018 Treasure Data Inc.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

```

Despite some licenses can be compatible with Apache, we want to keep things easy and clear avoiding a mix of Licenses across the project.

## Code review, no feelings

When we review your code submission, they must follow our coding style, the code should be clear enough, documented if required and the patch Subject and Description well formed (within others).

If your code needs some improvement, someone of the reviewers or core developers will write a comment in your Pull Request, so please take in count the suggestion there, otherwise your request will never be merged.

Despite the effort that took for you to create the contribution, that is not an indication that the code have to be merged into upstream, everything will be reviewed and must be aligned as the code base.