# Contributing to DAOS

Your contributions are most welcome! There are several good ways to suggest new features, offer to add a feature, or just begin a dialog about DAOS:

* Open an issue in [jira](https://jira.hpdd.intel.com)
* Suggest a feature, ask a question, start a discussion, etc. in our [user group](https://daos.groups.io/g/daos)
* Chat with members of the DAOS community real-time on [Gitter](https://gitter.im/daos-storage)

## Coding Rules

Please check the [coding conventions](https://wiki.hpdd.intel.com/display/DC/Coding+Rules) for code contribution.

## Commit Comments

### Commit Message Content

Writing good commit comments is critical to ensuring that changes are easily understood, even years after they were originally written. The commit comment should contain enough information about the change to allow the reader to understand the motivation for the change, what parts of the code it is affecting, and any interesting, unusual, or complex parts of the change to draw attention to.

The reason for a change may be manyfold: bug, enhancement, feature, code style, etc. so providing information about this sets the stage for understanding the change. If it is a bug, include information about what usage triggers the bug and how it manifests (error messages, assertion failure, etc.). If it is a feature, include information about what improvement is being made, and how it will affect usage.

Providing some high-level information about the code path that is being modified is useful for the reader, since the files and patch fragments are not necessarily going to be listed in a sensible order in the patch. Including the important functions being modified provides a starting point for the reader to follow the logic of the change, and makes it easier to search for such changes in the future.

If the patch is based on some earlier patch, then including the git commit hash of the original patch, Jira ticket number, etc. is useful for tracking the chain of dependencies. This can be very useful if a patch is landed separately to different maintenance branches, if it is fixing a problem in a previously landed patch, or if it is being imported from an upstream kernel commit.

Having long commit comments that describe the change well is a good thing. The commit comments will be tightly associated with the code for a long time into the future, even many of the original commit comments from years earlier are still available through changes of the source code repository. In contrast, bug tracking systems come and go, and cannot be relied upon to track information about a change for extended periods of time.

### Commit Message Format

Unlike the content of the commit message, the format is relatively easy to verify for correctness. Having the same standard format allows Git tools like git shortlog to extract information from the patches more easily.

The first line of the commit comment is the commit summary of the change. Changes submitted to the DAOS master branch require a DAOS Jira ticket number at the beginning of the commit summary. A DAOS Jira ticket is one that begins with DAOS and is, therefore, part of the DAOS project within Jira.

The commit summary should also have a component: tag immediately following the Jira ticket number that indicates to which DAOS subsystem the commit is related. Example DAOS subsystems relate to modules like client, pool, container, object, vos, rdb; functional components like rebuild; or auxiliary components like build, tests, doc. This subsystem list is not exhaustive but provides a good guideline for consistency.

The commit summary line must be 62 characters or less, including the Jira ticket number and component tag, so that git shortlog and git format-patch can fit the summary onto a single line. The summary must be followed by a blank line. The rest of the comments should be wrapped to 70 columns or less. This allows for the first line to be used as a subject in emails, and also for the entire body to be displayed using tools like git log or git shortlog in an 80 column window.

DAOS-nnn component: short description of change under 62 columns  
  
The "component:" should be a lower-case single-word subsystem of the  
DAOS code that best encompasses the change being made. Examples of  
components include modules: client, pool, container, object, vos, rdb,  
cart; functional subsystems: recovery; and auxiliary areas: build,  
tests, docs. This list is not exhaustive, but is a guideline.  
  
The commit comment should contain a detailed explanation of changes  
being made. This can be as long as you'd like. Please give details  
of what problem was solved (including error messages or problems that  
were seen), a good high-level description of how it was solved, and  
which parts of the code were changed (including important functions  
that were changed, if this is useful to understand the patch, and  
for easier searching). Wrap lines at/under 70 columns.  
  
Signed-off-by: Your Real Name <your\_email@domain.name>

### The Signed-off-by: line

The Signed-off-by: line asserts that you have permission to contribute the code to the project according to the Developer’s Certificate of Origin. The -s option to git commit also adds the Signed-off-by: line automatically.

### Additional commit tags

A number of additional commit tags can be used to further explain who has contributed to the patch, and for tracking purposes. These tags are commonly used with Linux kernel patches. These tags should appear before the Signed-off-by: tag.

Acked-by: User Name <user@domain.com>  
Tested-by: User Name <user@domain.com>  
Reported-by: User Name <user@domain.com>  
Reviewed-by: User Name <user@domain.com>  
CC: User Name <user@domain.com>

## Pull Requests (PR)

DAOS uses the common fork & merge workflow used by most GitHub-hosted projects. Please refer to the [online GitHub documentation](https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/proposing-changes-to-your-work-with-pull-requests).