# How to Contribute

The Deis project is Apache 2.0 licensed and accepts contributions via Github pull

requests. This document outlines some of the conventions on commit message formatting,

contact points for developers and other resources to make getting your contribution

accepted.

# Certificate of Origin

By contributing to this project you agree to the

[Developer Certificate of Origin (DCO)][dco]. This document was created by the Linux

Kernel community and is a simple statement that you, as a contributor, have the legal

right to make the contribution.

# Support Channels

Before opening a new issue, it's helpful to search the project - it's likely that another user

has already reported the issue you're facing, or it's a known issue that we're already aware of.

Additionally, see the [Troubleshooting Deis][troubleshooting] documentation for common issues.

Our official support channels are:

- GitHub issues: https://github.com/deis/deis/issues/new

- IRC: #[deis](irc://irc.freenode.org:6667/#deis) IRC channel on freenode.org

When opening an issue or asking for help on IRC, it's helpful to provide debug output from your cluster.

Run `/opt/bin/deis-debug-logs` on one of your CoreOS machines and provide the output to us in a [gist](https://gist.github.com/).

## Getting Started

- Fork the repository on GitHub

- Read [the documentation](http://docs.deis.io/en/latest/contributing/hacking/) for build instructions

## Contribution Flow

This is a rough outline of what a contributor's workflow looks like:

- Create a topic branch from where you want to base your work. This is usually master.

- Make commits of logical units.

- Make sure your commit messages are in the proper format, see below

- Push your changes to a topic branch in your fork of the repository.

- Submit a pull request

Thanks for your contributions!

### Design Documents

Most substantial changes to Deis should follow a [Design Document](http://docs.deis.io/en/latest/contributing/design-documents/)

describing the proposed changes and how they are tested and verified before they

are accepted into the project.

### Commit Style Guideline

We follow a rough convention for commit messages borrowed from CoreOS, who borrowed theirs

from AngularJS. This is an example of a commit:

feat(scripts/test-cluster): add a cluster test command

this uses tmux to setup a test cluster that you can easily kill and

start for debugging.

To make it more formal, it looks something like this:

{type}({scope}): {subject}

<BLANK LINE>

{body}

<BLANK LINE>

{footer}

The {scope} can be anything specifying place of the commit change.

The {subject} needs to use imperative, present tense: “change”, not “changed” nor

“changes”. The first letter should not be capitalized, and there is no dot (.) at the end.

Just like the {subject}, the message {body} needs to be in the present tense, and includes

the motivation for the change, as well as a contrast with the previous behavior. The first

letter in a paragraph must be capitalized.

All breaking changes need to be mentioned in the {footer} with the description of the

change, the justification behind the change and any migration notes required.

Any line of the commit message cannot be longer than 72 characters, with the subject line

limited to 50 characters. This allows the message to be easier to read on github as well

as in various git tools.

The allowed {types} are as follows:

feat -> feature

fix -> bug fix

docs -> documentation

style -> formatting

ref -> refactoring code

test -> adding missing tests

chore -> maintenance

### More Details on Commits

For more details see the [commit style guide][style-guide].

[dco]: DCO

[style-guide]: http://docs.deis.io/en/latest/contributing/standards/#commit-style-guide

[troubleshooting]: http://docs.deis.io/en/latest/troubleshooting\_deis/