How to Contribute

CoreOS projects are Apache 2.0 licensed and accept contributions via GitHub pull requests. This document outlines some of the conventions on development workflow, commit message formatting, contact points and other resources to make it easier to get your contribution accepted.

Certificate of Origin

By contributing to this project you agree to the Developer Certificate of Origin (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. See the DCO file for details.

Email and Chat

The project currently uses the general CoreOS email list and IRC channel:

- Email: coreos-dev
- IRC: #coreos IRC channel on freenode.org

Please avoid emailing maintainers found in the MAINTAINERS file directly. They are very busy and read the mailing lists.

Getting Started

- Fork the repository on GitHub
- Read the README for build and test instructions
- Play with the project, submit bugs, submit patches!

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 (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.
- Make sure the tests pass, and add any new tests as appropriate.
- Submit a pull request to the original repository.

Thanks for your contributions!

Format of the Commit Message

We follow a rough convention for commit messages that is designed to answer two questions: what changed and why. The subject line should feature the what and the body of the commit should describe the why.

```
scripts: add the test-cluster command

this uses tmux to setup a test cluster that you can easily kill and
start for debugging.

Fixes #38
```

The format can be described more formally as follows:

```
<subsystem>: <what changed>
<BLANK LINE>
<why this change was made>
<BLANK LINE>
<footer>
```

The first line is the subject and should be no longer than 70 characters, the second line is always blank, and other lines should be wrapped at 80 characters. This allows the message to be easier to read on GitHub as well as in various git tools.