# Contributing to Google Cloud Ruby Client

1. \*\*Sign one of the contributor license agreements below.\*\*

2. Fork the repo, develop and test your code changes.

3. Send a pull request.

## Contributor License Agreements

Before we can accept your pull requests you'll need to sign a Contributor License Agreement (CLA):

- \*\*If you are an individual writing original source code\*\* and \*\*you own the intellectual property\*\*, then you'll need to sign an [individual CLA](https://developers.google.com/open-source/cla/individual).

- \*\*If you work for a company that wants to allow you to contribute your work\*\*, then you'll need to sign a [corporate CLA](https://developers.google.com/open-source/cla/corporate).

You can sign these electronically (just scroll to the bottom). After that, we'll be able to accept your pull requests.

## Setup

In order to use the google-cloud-ruby console and run the project's tests, there is a

small amount of setup:

1. Install Ruby.

google-cloud-ruby requires Ruby 2.4+. You may choose to manage your Ruby and gem installations with [RVM](https://rvm.io/), [rbenv](https://github.com/rbenv/rbenv), or [chruby](https://github.com/postmodern/chruby).

2. Install [Bundler](http://bundler.io/).

```sh

$ gem install bundler

```

3. Install the top-level project dependencies.

```sh

$ bundle install

```

4. Install the dependencies for each package.

```sh

$ bundle exec rake bundleupdate

```

## Console

In order to run code interactively, you can automatically load google-cloud-ruby and

its dependencies in IRB with:

```sh

$ bundle exec rake console

```

## Tests

Tests are very important part of google-cloud-ruby. All contributions should include tests that ensure the contributed code behaves as expected.

To run the unit tests, documentation tests, and code style checks together for a package:

``` sh

$ cd <package-name>

$ bundle exec rake ci

```

To run the command above, plus all acceptance tests, use `rake ci:acceptance` or its handy alias, `rake ci:a`.

### Unit Tests

The project uses the [minitest](https://github.com/seattlerb/minitest) library, including [specs](https://github.com/seattlerb/minitest#specs), [mocks](https://github.com/seattlerb/minitest#mocks) and [minitest-autotest](https://github.com/seattlerb/minitest-autotest).

To run the unit tests for a package:

``` sh

$ cd <package-name>

$ bundle exec rake test

```

### Documentation Tests

The project tests the code examples in the each gem's [YARD]()-based documentation.

The example testing functions in a way that is very similar to unit testing, and in fact the library providing it, [yard-doctest](https://github.com/p0deje/yard-doctest), is based on the project's unit test library, [minitest](https://github.com/seattlerb/minitest).

To run the documentation tests for a package:

``` sh

$ cd <package-name>

$ bundle exec rake doctest

```

If you add, remove or modify documentation examples when working on a pull request, you may need to update the setup for the tests. The stubs and mocks required to run the tests are located in `support/doctest\_helper.rb` for each package. Please note that much of the setup is matched by the title of the [`@example`](http://www.rubydoc.info/gems/yard/file/docs/Tags.md#example) tag. If you alter an example's title, you may encounter breaking tests.

### Acceptance Tests

The google-cloud-ruby acceptance tests interact with the live service API (or APIs) for each package, including:

\* BigQuery

\* Cloud Datastore

\* Cloud DNS

\* Cloud Pub/Sub

\* Cloud Storage

Follow the instructions in the [Authentication guide](AUTHENTICATION.md) for enabling APIs. Some of the APIs may not yet be generally available, making it difficult for some contributors to successfully run the entire acceptance test suite. However, please ensure that you do successfully run acceptance tests for any code areas covered by your pull request.

To run the acceptance tests, first create and configure a project in the Google Developers Console, as described in the [Authentication guide](AUTHENTICATION.md). Be sure to download the JSON KEY file. Make note of the PROJECT\_ID and the KEYFILE location on your system.

#### Datastore acceptance tests

To run the Datastore acceptance tests, you must first create indexes used in the tests.

##### Datastore indexes

Install the [gcloud command-line tool](https://developers.google.com/cloud/sdk/gcloud/) and use it to create the indexes used in the datastore acceptance tests. From the project's root directory:

``` sh

# Install the app component

$ gcloud components update app

# Set the default project in your env

$ gcloud config set project PROJECT\_ID

# Authenticate the gcloud tool with your account

$ gcloud auth login

# Create the indexes

$ gcloud preview datastore create-indexes acceptance/data/

```

#### DNS Acceptance Tests

To run the DNS acceptance tests you must give your service account permissions to a domain name in [Webmaster Central](https://www.google.com/webmasters/verification) and set the `GCLOUD\_TEST\_DNS\_DOMAIN` environment variable to the fully qualified domain name. (e.g. "example.com.")

#### Running the acceptance tests

To run the acceptance tests for a package:

``` sh

$ cd <package-name>

$ bundle exec rake acceptance[PROJECT\_ID,KEYFILE\_PATH]

```

Or, if you prefer you can store the values in the `GCLOUD\_TEST\_PROJECT` and `GCLOUD\_TEST\_KEYFILE` environment variables:

``` sh

$ cd <package-name>

$ export GCLOUD\_TEST\_PROJECT=my-project-id

$ export GCLOUD\_TEST\_KEYFILE=/path/to/keyfile.json

$ bundle exec rake acceptance

```

If you want to use different values for Datastore vs. Storage acceptance tests, for example, you can use the `DATASTORE\_TEST\_` and `STORAGE\_TEST\_` environment variables:

``` sh

$ cd <package-name>

$ export DATASTORE\_TEST\_PROJECT=my-project-id

$ export DATASTORE\_TEST\_KEYFILE=/path/to/keyfile.json

$ export STORAGE\_TEST\_PROJECT=my-other-project-id

$ export STORAGE\_TEST\_KEYFILE=/path/to/other/keyfile.json

$ bundle exec rake acceptance

```

### Integration Tests

The google-cloud-ruby integration tests are end-to-end tests that validate library functionality on real Google Cloud Platform hosting environments. The integration process deploys several Rack-based applications to Google Cloud Platform one by one, then validates google-cloud-ruby code by making requests to these test applications.

See the [integration/README.md](integration/README.md) for instructions on how to setup for the integration tests and other details.

To run the integration tests:

```sh

$ bundle exec rake integration

```

## Coding Style

Please follow the established coding style in the library. The style is is largely based on [The Ruby Style Guide](https://github.com/bbatsov/ruby-style-guide) with a few exceptions based on seattle-style:

\* Avoid parenthesis when possible, including in method definitions.

\* Always use double quotes strings. ([Option B](https://github.com/bbatsov/ruby-style-guide#strings))

You can check your code against these rules by running Rubocop like so:

```sh

$ cd <package-name>

$ bundle exec rake rubocop

```

## Code of Conduct

Please note that this project is released with a Contributor Code of Conduct. By participating in this project you agree to abide by its terms. See [Code of Conduct](CODE\_OF\_CONDUCT.md) for more information.