# Contributing

I love pull requests from everyone! By the way, I have a favor to ask you with your contribution :bow:

## Reporting issue

- Please write your ...

- platform (macOS, Linux, Windows, ...)

- Ruby version

- Terraforming version

- Terraform version (if `terraform plan` failed)

- executed command and error message (if any)

## Making changes

- Currently, this gem supports only \_\_AWS\_\_ resources. Other providers are supported as separated gems.

- Datadog: [terraforming-datadog](https://github.com/dtan4/terraforming-datadog)

- DNSimple: [terraforming-dnsimple](https://github.com/dtan4/terraforming-dnsimple)

- Do not bump gem version in your pull request.

- Please follow the coding style of \_existing\_ code. Most of trivial rules can be checked by [RuboCop](https://github.com/bbatsov/rubocop) ([`rubocop.yml`](https://github.com/dtan4/terraforming/blob/master/.rubocop.yml)).

- Coding style is checked automatically by [SideCI](https://sideci.com) right after creating pull request. If there is error, SideCI comments at the point error occured.

- Please write tests for your changes. All tests are written with [RSpec](http://rspec.info/).

## Adding new resource

- Class name must match to Terraforming's resource name without `aws\_` prefix, and be a complete resource name.

- e.g. `aws\_iam\_group\_membership`: `IAMGroupMembership`

- Yes, I know that some of resources I added a long ago don't follow to this rule...

- File name must also match to Terraforming's resource name without `aws\_` prefix.

- e.g. `aws\_iam\_group\_membership`: `iam\_group\_membership.rb`

- Command name should be abbreviation.

- e.g. `aws\_iam\_group\_membership`: `iamgp`

- Please check generation result by executing `terraform plan` with real resources. There should be NO diff with generated `.tf` and `.tfstate`.

`script/generate` generates new resource code / test skeletons.

```bash

$ script/generate ec2

==> Generate ec2.rb

==> Generate ec2\_spec.rb

==> Generate ec2.erb

Add below code by hand.

lib/terraforming.rb:

require "terraforming/resource/ec2"

lib/terraforming/cli.rb:

module Terraforming

class CLI < Thor

# Subcommand name should be acronym.

desc "ec2", "Ec2"

def ec2

execute(Terraforming::Resource::Ec2, options)

end

spec/lib/terraforming/cli\_spec.rb:

module Terraforming

describe CLI do

context "resources" do

describe "ec2" do

let(:klass) { Terraforming::Resource::Ec2

let(:command) { :ec2 }

it\_behaves\_like "CLI examples"

end

```