Ruby Rules® for Bazel

Konstantin Gredeskoul

Table of Contents

> This repo is primarily maintained by Konstantin Gredeskoul and Yuki "Yugui" Sonoda. We are both very busy and would really love more contributors to join the core team. If you are interested in developing Ruby Rules for Bazel, please submit a couple of PRs and then lets talk!

Build Status & Activity

CI Status	Activity & Documentation
[CircleCI]	[activity]
[Build Status]	[changelog] [readme.pdf]

Chapter 1. Rules Development Status

Readiness	Types of Applications
Development Status Ready	ruby apps, ruby gems, micro-services, ideally in a mono-repo
Development Status Ready	medium-sized Ruby on Rails apps, ideally in a mono-repo
Development Status Wait	complex Ruby on Rails monoliths, single-repo

we have a short guide on Building your first Ruby Project on the Wiki. We encourage you to check it out.

Chapter 2. Table of Contents

Chapter 3. Usage

3.1. WORKSPACE File

3.1.1. Load dependencies, select Ruby SDK and define one or more Bundles

```
workspace(name = "my_ruby_project")

load("@bazel_tools//tools/build_defs/repo:http.bzl", "http_archive")
load("@bazel_tools//tools/build_defs/repo:git.bzl", "git_repository")

#------------------------------------------------------------------------
# To get the latest ruby rules, grab the 'master' branch.
#------------------------------------------------------------------------

git_repository(
    name = "bazelruby_rules_ruby",
    remote = "https://github.com/bazelruby/rules_ruby.git",
    branch = "master"
)

load(
    "@bazelruby_rules_ruby//ruby:deps.bzl",
    "rules_ruby_dependencies",
    "rules_ruby_select_sdk",
)

rules_ruby_dependencies()

#------------------------------------------------------------------------
# Specify Ruby version - this will either build Ruby or use a local
# RBENV installation if the Ruby version matches.
#------------------------------------------------------------------------

load("@bazel_skylib//:workspace.bzl", "bazel_skylib_workspace")
bazel_skylib_workspace()

rules_ruby_select_sdk(version = "3.0.1")

#------------------------------------------------------------------------
# Now, load the ruby_bundle rule & install gems specified in the Gemfile
#------------------------------------------------------------------------

load(
    "@bazelruby_rules_ruby//ruby:defs.bzl",
    "ruby_bundle",
)

ruby_bundle(
    name = "bundle",
    excludes = {
        "mini_portile": ["test/**/*"],
    },
    gemfile = "//:Gemfile",
    gemfile_lock = "//:Gemfile.lock",
)

# You can specify more than one bundle in the WORKSPACE file
ruby_bundle(
    name = "bundle_app_shopping",
    gemfile = "//apps/shopping:Gemfile",
    gemfile_lock = "//apps/shopping:Gemfile.lock",
)
```

3.2. BUILD.bazel file(s)

Any of the project `BUILD` files can now reference any gems included in the `Gemfile` referenced by the `ruby_bundle` rule, and defined in the project's `WORKSPACE` file.

3.2.1. Define Ruby Executable, Library and an RSpec

Add `ruby_library`, `ruby_binary`, `ruby_rspec` or `ruby_test` into your `BUILD.bazel` files.

```
#------------------------------------------------------------------------
# Define Ruby executable, test, spec and package a gem
#------------------------------------------------------------------------

load(
    "@bazelruby_rules_ruby//ruby:defs.bzl",
    "ruby_binary",
    "ruby_library",
    "ruby_test",
    "ruby_rspec",
)

ruby_library(
    name = "foo",
    srcs = glob(["lib/**/*.rb"]),
    includes = ["lib"],
    deps = [
      "@bundle//:activesupport",
      "@bundle//:awesome_print",
      "@bundle//:rubocop",
    ]
)

ruby_binary(
    name = "bar",
    srcs = ["bin/bar"],
    deps = [":foo"],
)

ruby_test(
    name = "foo-test",
    srcs = ["test/foo_test.rb"],
    deps = [":foo"],
)

ruby_rspec(
    name = "foo-spec",
    specs = glob(["spec/**/*.rb"]),
    rspec_args = { "--format": "progress" },
    deps = [":foo"]
}
```

3.2.2. Package Ruby files as a Gem

Use `ruby_gem` rule to package any number of ruby files or folders into a Ruby-Gem compatible ZIP archive.

```
load(
    "@bazelruby_rules_ruby//ruby:defs.bzl",
    "ruby_gem",
)

ruby_gem(
    name            = "awesome-sauce-gem", # name of the build target
    gem_name        = "awesome-sauce",     # name of the gem
    gem_version     = "0.1.0",
    gem_summary     = "Example gem to demonstrate Bazel Gem packaging",
    gem_description = "Example gem to demonstrate Bazel Gem packaging",
    gem_homepage    = "https://github.com/bazelruby/rules_ruby",
    gem_authors     = [
        "BazelRuby",
        "Konstantin Gredeskoul"
    ],
    gem_author_emails = [
        "bazelruby@googlegroups.com",
    ],
    gem_runtime_dependencies = {
        "colored2": "~> 3.1.2",
        "hashie": "",
    },
    gem_development_dependencies = {
        "rspec": "",
        "rspec-its": "",
        "rubocop": "",
    },
    srcs = [
        glob("{bin,exe,lib,spec}/**/*.rb")
    ],
    deps = [
        "//lib:example_gem",
    ],
)
```

3.3. Tool Specific Setup

3.3.1. ASDF

If you are using ASDF to manage your ruby installs, you can use them by adding `.bazelrc`:

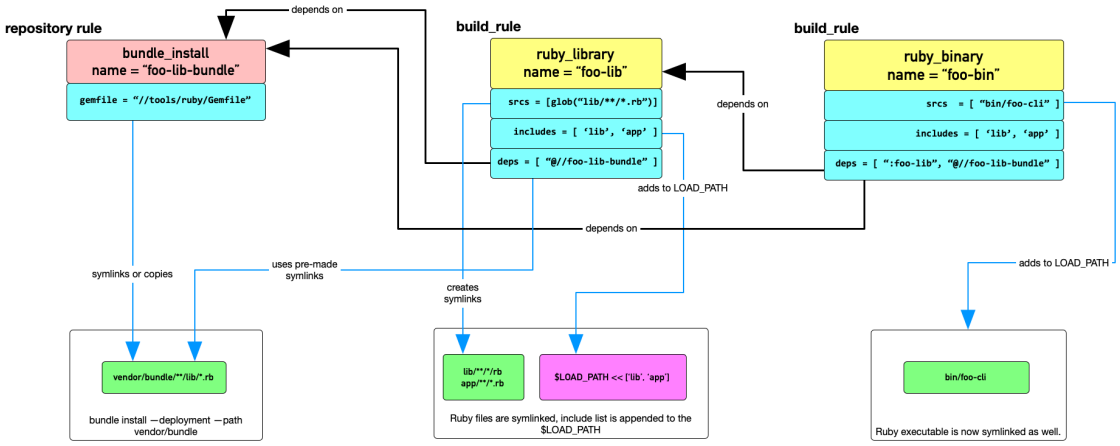
```
build --test_env=ASDF_DIR --test_env=ASDF_DATA_DIR
build --action_env=ASDF_DIR --test_env=ASDF_DATA_DIR
```

You will have to be sure to export the `ASDF_DATA_DIR` in your profile since it's not set by default. e.g. `export ASDF_DATA_DIR="$HOME/.asdf"`

3.4. Rule Dependency Diagram

this diagram is slightly outdated.

The following diagram attempts to capture the implementation behind `ruby_library` that depends on the result of `bundle install`, and a `ruby_binary` that depends on both:



Ruby Bazell Rules, their actions and dependencies

Chapter 4. Rules

4.1. ruby_library

```
ruby_library(
    name,
    deps,
    srcs,
    data,
    compatible_with,
    deprecation,
    distribs,
    features,
    licenses,
    restricted_to,
    tags,
    testonly,
    toolchains,
    visibility)
```

<table class="table table-condensed table-bordered table-params"><colgroup><col class="col-param"></col> <col class="param-description"></col></colgroup> <thead><tr><th colspan="2">Attributes</th></tr></thead> <tbody><tr><td><code>name</code></td> <td><code>Name, required</code> A unique name for this rule. </td></tr> <tr><td><code>srcs</code></td> <td><code>List of Labels, optional</code> List of <code>.rb</code> files. At least <code>srcs</code> or <code>deps</code> must be present </td></tr> <tr><td><code>deps</code></td> <td><code>List of labels, optional</code> List of targets that are required by the <code>srcs</code> Ruby files. At least <code>srcs</code> or <code>deps</code> must be present </td></tr> <tr><td><code>includes</code></td> <td><code>List of strings, optional</code> List of paths to be added to <code>$LOAD_PATH</code> at runtime. The paths must be relative to the the workspace which this rule belongs to. </td></tr> <tr><td><code>rubyopt</code></td> <td><code>List of strings, optional</code> List of options to be passed to the Ruby interpreter at runtime. NOTE: <code>-I</code> option should usually go to <code>includes</code> attribute. </td></tr></tbody> <tbody><tr><td colspan="2">And other common attributes</td></tr></tbody></table>

4.2. ruby_binary

```
ruby_binary(
    name,
    deps,
    srcs,
    data,
    main,
    compatible_with,
    deprecation,
    distribs,
    features,
    licenses,
    restricted_to,
    tags,
    testonly,
    toolchains,
    visibility,
    args,
    output_licenses
)
```

<table class="table table-condensed table-bordered table-params"><colgroup><col class="col-param"></col> <col class="param-description"></col></colgroup> <thead><tr><th colspan="2">Attributes</th></tr></thead> <tbody><tr><td><code>name</code></td> <td><code>Name, required</code> A unique name for this rule. </td></tr> <tr><td><code>srcs</code></td> <td><code>List of Labels, required</code> List of <code>.rb</code> files. </td></tr> <tr><td><code>deps</code></td> <td><code>List of labels, optional</code> List of targets that are required by the <code>srcs</code> Ruby files. </td></tr> <tr><td><code>main</code></td> <td><code>Label, optional</code> The entrypoint file. It must be also in <code>srcs</code>. If not specified, <code><var>$(NAME)</var>.rb</code> where <code>$(NAME)</code> is the <code>name</code> of this rule. </td></tr> <tr><td><code>includes</code></td> <td><code>List of strings, optional</code> List of paths to be added to <code>$LOAD_PATH</code> at runtime. The paths must be relative to the the workspace which this rule belongs to. </td></tr> <tr><td><code>rubyopt</code></td> <td><code>List of strings, optional</code> List of options to be passed to the Ruby interpreter at runtime. NOTE: <code>-I</code> option should usually go to <code>includes</code> attribute. </td></tr></tbody> <tbody><tr><td colspan="2">And other common attributes</td></tr></tbody></table>

4.3. ruby_test

```
ruby_test(
    name,
    deps,
    srcs,
    data,
    main,
    compatible_with,
    deprecation,
    distribs,
    features,
    licenses,
    restricted_to,
    tags,
    testonly,
    toolchains,
    visibility,
    args,
    size,
    timeout,
    flaky,
    local,
    shard_count
)
```

<table class="table table-condensed table-bordered table-params"><colgroup><col class="col-param"></col> <col class="param-description"></col></colgroup> <thead><tr><th colspan="2">Attributes</th></tr></thead> <tbody><tr><td><code>name</code></td> <td><code>Name, required</code> A unique name for this rule. </td></tr> <tr><td><code>srcs</code></td> <td><code>List of Labels, required</code> List of <code>.rb</code> files. </td></tr> <tr><td><code>deps</code></td> <td><code>List of labels, optional</code> List of targets that are required by the <code>srcs</code> Ruby files. </td></tr> <tr><td><code>main</code></td> <td><code>Label, optional</code> The entrypoint file. It must be also in <code>srcs</code>. If not specified, <code><var>$(NAME)</var>.rb</code> where <code>$(NAME)</code> is the <code>name</code> of this rule. </td></tr> <tr><td><code>includes</code></td> <td><code>List of strings, optional</code> List of paths to be added to <code>$LOAD_PATH</code> at runtime. The paths must be relative to the the workspace which this rule belongs to. </td></tr> <tr><td><code>rubyopt</code></td> <td><code>List of strings, optional</code> List of options to be passed to the Ruby interpreter at runtime. NOTE: <code>-I</code> option should usually go to <code>includes</code> attribute. </td></tr></tbody> <tbody><tr><td colspan="2">And other common attributes</td></tr></tbody></table>

4.4. ruby_bundle

NOTE: This is a repository rule, and can only be used in a `WORKSPACE` file.

This rule installs gems defined in a Gemfile using Bundler, and exports individual gems from the bundle, as well as the entire bundle, available as a `ruby_library` that can be depended upon from other targets.

```
ruby_bundle(
    name,
    gemfile,
    gemfile_lock,
    bundler_version = "2.1.4",
    excludes = [],
    ruby_sdk = "@org_ruby_lang_ruby_toolchain",
    ruby_interpreter = "@org_ruby_lang_ruby_toolchain//:ruby",
)
```

<table class="table table-condensed table-bordered table-params"><colgroup><col class="col-param"></col> <col class="param-description"></col></colgroup> <thead><tr><th colspan="2">Attributes</th></tr></thead> <tbody><tr><td><code>name</code></td> <td><code>Name, required</code> A unique name for this rule. </td></tr> <tr><td><code>gemfile</code></td> <td><code>Label, required</code> The <code>Gemfile</code> which Bundler runs with. </td></tr> <tr><td><code>gemfile_lock</code></td> <td><code>Label, required</code> The <code>Gemfile.lock</code> which Bundler runs with. NOTE: This rule never updates the <code>Gemfile.lock</code>. It is your responsibility to generate/update <code>Gemfile.lock</code> </td></tr> <tr><td><code>bundler_version</code></td> <td><code>String, optional</code> The Version of Bundler to use. Defaults to 2.1.4. NOTE: This rule never updates the <code>Gemfile.lock</code>. It is your responsibility to generate/update <code>Gemfile.lock</code> </td></tr></tbody></table>

4.4.1. Limitations

Installing using a `Gemfile` that uses the `gemspec` keyword is not currently supported.

4.4.2. Conventions

`ruby_bundle` creates several targets that can be used downstream. In the examples below we assume that your `ruby_bundle` has a name `app_bundle`:

- `@app_bundle//:bundler` — references just the Bundler from the bundle.
- `@app_bundle//:gems` — references *all* gems in the bundle (i.e. "the entire bundle").
- `@app_bundle//:gem-name` — references *just the specified* gem in the bundle, eg. `@app_bundle//:awesome_print`.
- `@app_bundle//:bin` — references to all installed executables from this bundle, with individual executables accessible via eg. `@app_bundle//:bin/rubocop`

4.4.3. WORKSPACE:

```
load("@bazelruby_rules_ruby//ruby:defs.bzl", "ruby_bundle")

ruby_bundle(
    name = "gems",
    bundler_version = '2.1.4',
    gemfile = "//:Gemfile",
    gemfile_lock = "//:Gemfile.lock",
)
```

4.4.4. BUILD.bazel:

```
# Reference the entire bundle with :gems

ruby_library(
    name = "foo",
    srcs = ["foo.rb"],
    deps = ["@gems//:gems"],
)

# Or, reference specific gems from the bundle like so:

ruby_binary(
    name = "rubocop",
    srcs = [":foo", ".rubocop.yml"],
    args = ["-P", "-D", "-c" ".rubocop.yml"],
    main = "@gems//:bin/rubocop",
    deps = ["@gems//:rubocop"],
)
```

4.5. ruby_rspec

```
ruby_rspec(
    name,
    deps,
    srcs,
    data,
    main,
    rspec_args,
    bundle,
    compatible_with,
    deprecation,
    distribs,
    features,
    licenses,
    restricted_to,
    tags,
    testonly,
    toolchains,
    visibility,
    args,
    size,
    timeout,
    flaky,
    local,
    shard_count
)
```

<table class="table table-condensed table-bordered table-params"><colgroup><col class="col-param"></col> <col class="param-description"></col></colgroup> <thead><tr><th colspan="2">Attributes</th></tr></thead> <tbody><tr><td><code>name</code></td> <td><code>Name, required</code> A unique name for this rule. </td></tr> <tr><td><code>srcs</code></td> <td><code>List of Labels, required</code> List of <code>.rb</code> files. </td></tr> <tr><td><code>deps</code></td> <td><code>List of labels, optional</code> List of targets that are required by the <code>srcs</code> Ruby files. </td></tr> <tr><td><code>main</code></td> <td><code>Label, optional</code> The entrypoint file. It must be also in <code>srcs</code>. If not specified, <code><var>$(NAME)</var>.rb</code> where <code>$(NAME)</code> is the <code>name</code> of this rule. </td></tr> <tr><td><code>rspec_args</code></td> <td><code>List of strings, optional</code> Command line arguments to the <code>rspec</code> binary, eg <code><span class=""--progress""></code> If not specified, the default arguments defined in <code>constants.bzl</code> are used: <code>--format=documentation --force -color</code>. </td></tr> <tr><td><code>includes</code></td> <td><code>List of strings, optional</code> List of paths to be added to <code>$LOAD_PATH</code> at runtime. The paths must be relative to the the workspace which this rule belongs to. </td></tr> <tr><td><code>rubyopt</code></td> <td><code>List of strings, optional</code> List of options to be passed to the Ruby interpreter at runtime. NOTE: <code>-I</code> option should usually go to <code>includes</code> attribute. </td></tr></tbody> <tbody><tr><td colspan="2">And other common attributes</td></tr></tbody></table>

4.6. ruby_gem

Used to generate a zipped gem containing its srcs, dependencies and a gemspec.

```
ruby_gem(
    name,
    gem_name,
    gem_version,
    gem_summary,
    gem_description,
    gem_homepage,
    gem_authors,
    gem_author_emails,
    gem_runtime_dependencies,
    gem_development_dependencies,
    require_paths = ["lib"],
    srcs = srcs,
    deps = deps,
    data = data
)
```

<table class="table table-condensed table-bordered table-params"><colgroup><col class="col-param"></col> <col class="param-description"></col></colgroup> <thead><tr><th colspan="2">Attributes</th></tr></thead> <tbody><tr><td><code>name</code></td> <td><code>Name, required</code> A unique name for this build target. </td></tr> <tr><td><code>gem_name</code></td> <td><code>Name of the gem, required</code> The name of the gem to be generated. </td></tr> <tr><td><code>gem_version</code></td> <td><code>String, optional</code> The version of the gem. Is used to name the output file, which becomes <code>name-version.zip</code>, and also included in the Gemspec. </td></tr> <tr><td><code>gem_summary</code></td> <td><code>String, optional</code> One line summary of the gem purpose. </td></tr> <tr><td><code>gem_description</code></td> <td><code>String, required</code> Single-line, paragraph-sized description text for the gem. </td></tr> <tr><td><code>gem_homepage</code></td> <td><code>String, optional</code> Homepage URL of the gem. </td></tr> <tr><td><code>gem_authors</code></td> <td><code>List of Strings, required</code> List of human readable names of the gem authors. Required to generate a valid gemspec. </td></tr> <tr><td><code>gem_author_emails</code></td> <td><code>List of Strings, optional</code> List of email addresses of the authors. </td></tr> <tr><td><code>srcs</code></td> <td><code>List of Labels, optional</code> List of <code>.rb</code> files. At least <code>srcs</code> or <code>deps</code> must be present </td></tr> <tr><td><code>deps</code></td> <td><code>List of labels, optional</code> List of targets that are required by the <code>srcs</code> Ruby files. At least <code>srcs</code> or <code>deps</code> must be present </td></tr> <tr><td><code>require_paths</code></td> <td><code>List of Strings, optional</code> List of paths to be added to the Ruby LOAD_PATH when using this gem. Typically this value is just <code>lib</code> (which is also the default). </td></tr> <tr><td><code>gem_runtime_dependencies</code></td> <td><code>String Dictionary, optional</code> This is a dictionary where keys are gem names, and values are either an empty string or a gem version specification. For instance, the pessimistic version specifier <code>~> 3.0</code> means that all versions up to <code>4.0</code> are accepted. </td></tr> <tr><td><code>gem_development_dependencies</code></td> <td><code>String Dictionary, optional</code> Similar to the above, this specifies gems necessary for the development of the above gem, such as testing gems, linters, code coverage and more. </td></tr></tbody></table>

Chapter 5. Potential Future Features

- <input type="checkbox" class="task-list-item-checkbox" disabled="disabled" checked="checked"></input>Using various versions of Ruby installed locally
- <input type="checkbox" class="task-list-item-checkbox" disabled="disabled"></input>Building native extensions in gems with Bazel
- <input type="checkbox" class="task-list-item-checkbox" disabled="disabled"></input>Releasing your gems with Bazel (Coinbase fork might have this feature, worth checking)

Chapter 6. Contributing

We welcome contributions to RulesRuby. Please make yourself familiar with the code of conduct, which basically says — don't be an a-hole.

You may notice that there is more than one Bazel WORKSPACE inside this repo. There is one in `examples/simple_script` for instance, because we use this example to validate and test the rules. So be mindful whether your current directory contains `WORKSPACE` file or not.

6.1. Setup

6.1.1. Using the Script

You will need Homebrew installed prior to running the script.

After that, cd into the top level folder and run the setup script in your Terminal:

```
› bin/setup
```

This runs a complete setup, shouldn't take too long. You can explore various script options with the `help` command:

```
› bin/setup help
USAGE
  # without any arguments runs a complete setup.
  bin/setup

  # alternatively, a sub-setup function name can be passed:
  bin/setup [ gems | git-hook | help | os-specific | main | remove-git-hook ]

DESCRIPTION:
  Runs full setup without any arguments.

  Accepts one optional argument - one of the actions that typically run
  as part of setup, with one exception - remove-git-hook.
  This action removes the git commit hook installed by the setup.

EXAMPLES:
    bin/setup - runs the entire setup.
```

6.1.2. OS-Specific Setup

Note that the setup contains `os-specific` section. This is because there are two extension scripts:

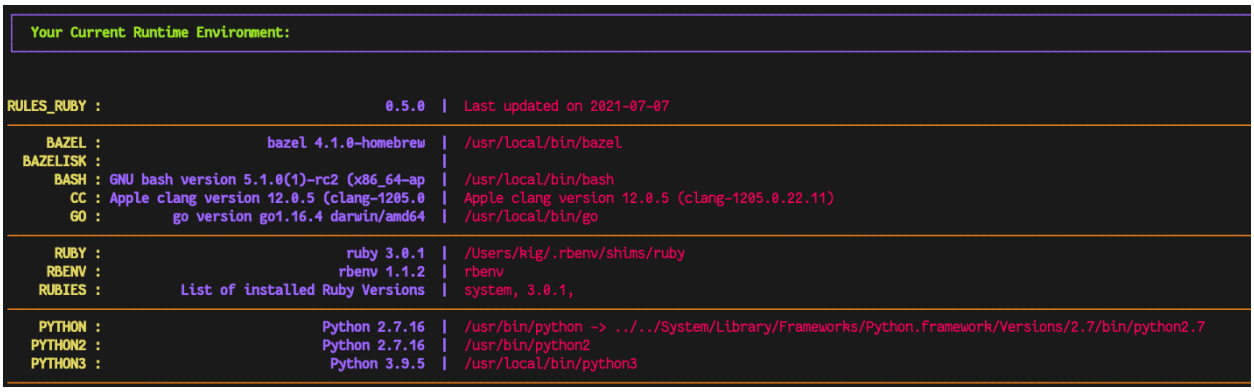
- `bin/setup-linux`
- `bin/setup-darwin`

Those will install Bazel and everything else you need on either platform. In fact, we use the linux version on CI.

6.2. Verifying Your Environment

We provided a handy script `bin/show-env` to display where your dependencies are coming from. Here is an example of running it on a Mac OS-X system:

```
> bin/show-env
```



6.2.1. Issues During Setup

> **Please report any errors to `bin/setup` as Issues on Github. You can assign them to @kigster.** If I am not responding fast enough, and you are in a hurry, please email kigster AT gmail directly.

6.3. Developing Rules

Besides making yourself familiar with the existing code, and Bazel documentation on writing rules, you might want to follow this order:

1. Setup dev tools as described in the setup section.
2. hack, hack, hack...
3. Make sure all tests pass — you can run a single command for that (but see more on it below.

```
bin/test-suite
```

OR, you can run individual Bazel test commands from the inside.

- `bazel test //...`
- `cd examples/simple_script && bazel test //...`
    1. Open a pull request in Github, and please be as verbose as possible in your description.

In general, it's always a good idea to ask questions first — you can do so by creating an issue.

6.4. Running Tests

After running setup, and since this is a bazel repo you can use Bazel commands:

```
bazel build //...:all
bazel query //...:all
bazel test  //...:all
```

But to run tests inside each sub-WORKSPACE, you will need to repeat that in each sub-folder. Luckily, there is a better way.

6.4.1. Test Script

This script runs all tests (including sub-workspaces) when ran without arguments:

```
bin/test-suite
```

Run it with `help` command to see other options, and to see what parts you can run individually. At the moment they are:

```
# alternatively, a partial test name can be passed:
bin/test-suite [ all | bazel-info | buildifier | help | rspec | rubocop | simple-script |  workspace ]
```

On a MacBook Pro it takes about 3 minutes to run.

6.5. Linter

We are using RuboCop for ruby and Buildifier for Bazel. Both are represented by a single script `bin/linter`, which just like the scripts above runs ALL linters when ran without arguments, accepts `help` commnd, and can be run on a subset of linting strategies:

```
bin/linter
```

The following are the partial linting functions you can run:

```
# alternatively, a partial linter name can be passed:
bin/linter [ all | buildifier | help | rubocop ]
```

6.6. Regenerating README.pdf & Changelog

To regenerate, first you may need to grab an API token and export the `GITHUB_TOKEN` variable:

```
export GITHUB_TOKEN=....
```

Then use the `make` target:

```
make update
```

Or, manually:

```
gem install github_changelog_generator
github_changelog_generator -u bazelruby -p rules_ruby -t your-github-token
```

Chapter 7. Copyright

Core Team:

- Yuki Yugui Sonoda
- Konstantin Gredeskoul

Core Team (Emeritus):

- Graham Jenson