








 **lorenzo** attempt to fix docker building in release workflow ✓

28fc39f · 3 months ago 1,174 Commits


 .github	attempt to fix docker building ...	3 months ago
 app	Overhaul: Reorganize tests, re...	4 years ago
 contrib	delete rule DL3005 (#964)	2 years ago
 docker	Add LABEL org.opencontainer...	2 years ago
 docs	Spelling (#936)	2 years ago
 scripts	Lint all markdown files by stric...	8 years ago
 src	- remove haskell.nix	3 months ago
 test	Pragma: Allow comments follo...	3 months ago
 .dockerignore	Using a single Dockefile for th...	7 years ago
 .envrc	- remove haskell.nix	3 months ago
 .gitignore	- remove haskell.nix	3 months ago
 .hindent.yaml	Lint with hindent to improve r...	8 years ago
	unpin docker version in pre-c...	3 years ago

Dockerfile linter, validate inline bash, written in Haskell













#docker #dockerfile #haskell #linter
#static-analysis #shellcheck #dockerfile-linter

-  Readme
 -  GPL-3.0 license
 -  Activity
 -  Custom properties
 -  11.2k stars
 -  72 watching
 -  448 forks
- Report repository


Releases 79

 **v2.12.0** Latest
on Nov 10, 2022

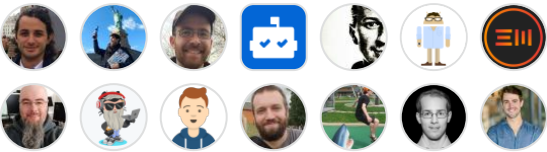
+ 78 releases

 .pre-commit-hooks.yaml	Add MD linter configuration	7 years ago
 LICENSE	Must change license to GPL	10 years ago
 README.md	docs: Update TOC with Ignori...	last year
 Setup.hs	Port all tests over to HSpec	9 years ago
 ThirdPartyNotices.txt	Update ThirdPartyNotices.txt	3 years ago
 cabal.project	- remove haskell.nix	3 months ago
 devenv.lock	- remove haskell.nix	3 months ago
 devenv.nix	- remove haskell.nix	3 months ago
 devenv.yaml	- remove haskell.nix	3 months ago
 hadolint.cabal	remove static flag	3 months ago
 integration_test.sh	Spelling (#936)	2 years ago
 screenshot.png	[ImgBot] Optimize images	6 years ago

Packages 1

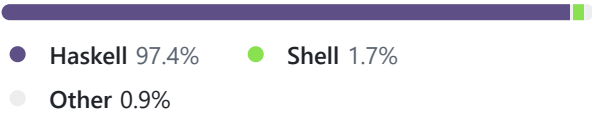
 hadolint

Contributors 120



[+ 106 contributors](#)

Languages



Haskell Dockerfile Linter



A smarter Dockerfile linter that helps you build [best practice](#) Docker images. The linter parses the Dockerfile into an AST and performs rules on top of the AST. It stands on the shoulders of [ShellCheck](#) to lint the Bash code inside `RUN` instructions.

[Check the online version on hadolint.github.io/hadolint](#)

```
DL4000 Specify a maintainer of the Dockerfile
DL3006 Always tag the version of an image explicitly.
1 FROM debian
SC1007 Remove space after = if trying to assign a value (for empty string, use var='' ... ).
SC2154 node_version is referenced but not assigned.
DL3009 Delete the apt-get lists after installing something
2 RUN node_version= "0.10" \
3   && apt-get update && apt-get -y install nodejs="$node_version"
4 COPY package.json usr/src/app
DL3003 Use WORKDIR to switch to a directory
5 RUN cd /usr/src/app \
6   && npm install node-static
7
DL3011 Valid UNIX ports range from 0 to 65535
8 EXPOSE 80000
9 CMD ["npm", "start"]
```

Table of Contents

- [How to use](#)
- [Install](#)
- [CLI](#)
- [Configure](#)

- [Non-Posix Shells](#)
- [Ignoring Rules](#)
 - [Inline ignores](#)
 - [Global ignores](#)
- [Linting Labels](#)
 - [Note on dealing with variables in labels](#)
- [Integrations](#)
- [Rules](#)
- [Develop](#)
 - [Setup](#)
 - [REPL](#)
 - [Tests](#)
 - [AST](#)
 - [Building against custom libraries](#)
- [Alternatives](#)

How to use

You can run `hadolint` locally to lint your Dockerfile.

```
hadolint <Dockerfile>
hadolint --ignore DL3003 --ignore DL3006 <Dockerfile> # exclude specific rules
hadolint --trusted-registry my-company.com:500 <Dockerfile> # Warn when using untrusted FROM images
```



Docker comes to the rescue, providing an easy way how to run `hadolint` on most platforms. Just pipe your `Dockerfile` to `docker run`:

```
docker run --rm -i hadolint/hadolint < Dockerfile
# OR
docker run --rm -i ghcr.io/hadolint/hadolint < Dockerfile
```



or using [Podman](#):

```
podman run --rm -i docker.io/hadolint/hadolint < Dockerfile
# OR
podman run --rm -i ghcr.io/hadolint/hadolint < Dockerfile
```

or using Windows PowerShell:

```
cat .\Dockerfile | docker run --rm -i hadolint/hadolint
```

Install

You can download prebuilt binaries for OSX, Windows and Linux from the latest [release page](#). However, if this does not work for you, please fall back to container (Docker), `brew` or source installation.

On OSX, you can use [brew](#) to install `hadolint`.

```
brew install hadolint
```

On Windows, you can use [scoop](#) to install `hadolint`.

```
scoop install hadolint
```

On distributions that have `nix` installed, you can use the `hadolint` package to run ad-hoc shells or permanently install `hadolint` into your environment.

As mentioned earlier, `hadolint` is available as a container image:

```
docker pull hadolint/hadolint
# OR
docker pull ghcr.io/hadolint/hadolint
```

If you need a container with shell access, use the Debian or Alpine variants:

```
docker pull hadolint/hadolint:latest-debian
# OR
docker pull hadolint/hadolint:latest-alpine
# OR
docker pull ghcr.io/hadolint/hadolint:latest-debian
# OR
docker pull ghcr.io/hadolint/hadolint:latest-alpine
```

You can also build `hadolint` locally. You need [Haskell](#) and the [cabal](#) build tool to build the binary.

```
git clone https://github.com/hadolint/hadolint \
  && cd hadolint \
  && cabal configure \
  && cabal build \
  && cabal install
```

If you want the [VS Code Hadolint](#) extension to use Hadolint in a container, you can use the following [wrapper script](#):

```
#!/bin/bash
dockerfile="$1"
shift
docker run --rm -i hadolint/hadolint hadolint "$@" - < "$dockerfile"
```

CLI

```
hadolint --help
```

```
hadolint - Dockerfile Linter written in Haskell
```

```
Usage: hadolint [-v|--version] [-c|--config FILENAME] [DOCKERFILE...]
```

```
[--file-path-in-report FILEPATHINREPORT] [--no-fail]
[--no-color] [-V|--verbose] [-f|--format ARG] [--error RULECODE]
[--warning RULECODE] [--info RULECODE] [--style RULECODE]
[--ignore RULECODE]
[--trusted-registry REGISTRY (e.g. docker.io)]
[--require-label LABELSCHEMA (e.g. maintainer:text)]
[--strict-labels] [--disable-ignore-pragma]
[-t|--failure-threshold THRESHOLD]
```

Lint Dockerfile for errors and best practices

Available options:

-h,--help	Show this help text
-v,--version	Show version
-c,--config FILENAME	Path to the configuration file
--file-path-in-report FILEPATHINREPORT	The file path referenced in the generated report. This only applies for the 'checkstyle' format and is useful when running Hadolint with Docker to set the correct file path.
--no-fail	Don't exit with a failure status code when any rule is violated
--no-color	Don't colorize output
-V,--verbose	Enables verbose logging of hadolint's output to stderr
-f,--format ARG	The output format for the results [tty json checkstyle codeclimate gitlab_codeclimate gnu codacy sonarqube sarif] (default: tty)
--error RULECODE	Make the rule `RULECODE` have the level `error`
--warning RULECODE	Make the rule `RULECODE` have the level `warning`
--info RULECODE	Make the rule `RULECODE` have the level `info`
--style RULECODE	Make the rule `RULECODE` have the level `style`
--ignore RULECODE	A rule to ignore. If present, the ignore list in the config file is ignored
--trusted-registry REGISTRY (e.g. docker.io)	A docker registry to allow to appear in FROM instructions
--require-label LABELSCHEMA (e.g. maintainer:text)	The option --require-label=label:format makes Hadolint check that the label `label` conforms to format requirement `format`

```

--strict-labels          Do not permit labels other than specified in
                        `label-schema`
--disable-ignore-pragma  Disable inline ignore pragmas `# hadolint
                        ignore=DLxxxx`
-t,--failure-threshold THRESHOLD
                        Exit with failure code only when rules with a
                        severity equal to or above THRESHOLD are violated.
                        Accepted values: [error | warning | info | style |
                        ignore | none] (default: info)

```

Configure

Configuration files can be used globally or per project. Hadolint looks for configuration files in the following locations or their platform specific equivalents in this order and uses the first one exclusively:

- `$PWD/.hadolint.yaml`
- `$XDG_CONFIG_HOME/hadolint.yaml`
- `$HOME/.config/hadolint.yaml`
- `$HOME/.hadolint/hadolint.yaml` or `$HOME/hadolint/config.yaml`
- `$HOME/.hadolint.yaml`

In windows, the `%LOCALAPPDATA%` environment variable is used instead of `XDG_CONFIG_HOME`. Config files can have either `yaml` or `ym1` extensions.

`hadolint` full `yaml` config file schema

```

failure-threshold: string      # name of threshold level (error | warning | info | style | ignore | none)
format: string                 # Output format (tty | json | checkstyle | codeclimate | gitlab_codeclimate | gnu | code)
ignored: [string]              # list of rules
label-schema:                  # See Linting Labels below for specific label-schema details
  author: string                # Your name
  contact: string               # email address
  created: timestamp            # rfc3339 datetime
  version: string               # semver
  documentation: string         # url

```




```
git-revision: string      # hash
license: string           # spdx
no-color: boolean         # true | false
no-fail: boolean          # true | false
override:
  error: [string]         # list of rules
  warning: [string]       # list of rules
  info: [string]          # list of rules
  style: [string]         # list of rules
strict-labels: boolean    # true | false
disable-ignore-pragma: boolean # true | false
trustedRegistries: string | [string] # registry or list of registries
```

`hadolint` supports specifying the ignored rules using a configuration file. The configuration file should be in `yaml` format. This is one valid configuration file as an example:

```
ignored:
  - DL3000
  - SC1010
```

Additionally, `hadolint` can warn you when images from untrusted repositories are being used in Dockerfiles, you can append the `trustedRegistries` keys to the configuration file, as shown below:

```
ignored:
  - DL3000
  - SC1010

trustedRegistries:
  - docker.io
  - my-company.com:5000
  - "*.gcr.io"
```

If you want to override the severity of specific rules, you can do that too:

```
error:
- DL3001
- DL3002
warning:
- DL3042
- DL3033
info:
- DL3032
style:
- DL3015
```

`failure-threshold` Exit with failure code only when rules with a severity above THRESHOLD are violated (Available in v2.6.0+)

```
failure-threshold: info
override:
warning:
- DL3042
- DL3033
info:
- DL3032
```

Additionally, you can pass a custom configuration file in the command line with the `--config` option

```
hadolint --config /path/to/config.yaml Dockerfile
```

To pass a custom configuration file (using relative or absolute path) to a container, use the following command:

```
docker run --rm -i -v /your/path/to/hadolint.yaml:/.config/hadolint.yaml hadolint/hadolint < Dockerfile
# OR
docker run --rm -i -v /your/path/to/hadolint.yaml:/.config/hadolint.yaml ghcr.io/hadolint/hadolint < Dockerfile
```

In addition to config files, Hadolint can be configured with environment variables.

```
NO_COLOR=1 # Set or unset. See https://no-color.org
HADOLINT_NOFAIL=1 # Truthy value e.g. 1, true or yes
HADOLINT_VERBOSE=1 # Truthy value e.g. 1, true or yes
HADOLINT_FORMAT=json # Output format (tty | json | checkstyle | codeclimate | gitlab_codeclimate | gnu | cor
HADOLINT_FAILURE_THRESHOLD=info # threshold level (error | warning | info | style | ignore | none)
HADOLINT_OVERRIDE_ERROR=DL3010,DL3020 # comma separated list of rule codes
HADOLINT_OVERRIDE_WARNING=DL3010,DL3020 # comma separated list of rule codes
HADOLINT_OVERRIDE_INFO=DL3010,DL3020 # comma separated list of rule codes
HADOLINT_OVERRIDE_STYLE=DL3010,DL3020 # comma separated list of rule codes
HADOLINT_IGNORE=DL3010,DL3020 # comma separated list of rule codes
HADOLINT_STRICT_LABELS=1 # Truthy value e.g. 1, true or yes
HADOLINT_DISABLE_IGNORE_PRAGMA=1 # Truthy value e.g. 1, true or yes
HADOLINT_TRUSTED_REGISTRIES=docker.io # comma separated list of registry urls
HADOLINT_REQUIRE_LABELS=maintainer:text # comma separated list of label schema items
```

Non-Posix Shells

When using base images with non-posix shells as default (e.g. Windows based images) a special pragma `hadolint shell` can specify which shell the base image uses, so that Hadolint can automatically ignore all shell-specific rules.

```
FROM mcr.microsoft.com/windows/servercore:ltsc2022
# hadolint shell=powershell
RUN Get-Process notepad | Stop-Process
```

Ignoring Rules

Inline ignores

It is also possible to ignore rules by adding a special comment directly above the Dockerfile statement for which you want to make an exception for. Such comments look like `# hadolint ignore=DL3001,SC1081`. For example:

```
# hadolint ignore=DL3006
```

```
FROM ubuntu
```

```
# hadolint ignore=DL3003,SC1035
```

```
RUN cd /tmp && echo "hello!"
```



The comment "inline ignores" applies only to the statement following it.

Global ignores

Rules can also be ignored on a per-file basis using the global ignore pragma. It works just like inline ignores, except that it applies to the whole file instead of just the next line.

```
# hadolint global ignore=DL3003,DL3006,SC1035
```

```
FROM ubuntu
```

```
RUN cd /tmp && echo "foo"
```



Linting Labels

Hadolint is able to check if specific labels are present and conform to a predefined label schema. First, a label schema must be defined either via the command line:

```
hadolint --require-label author:text --require-label version:semver Dockerfile
```



or via the config file:

```
label-schema:  
  author: text  
  contact: email  
  created: rfc3339  
  version: semver
```



```
documentation: url
git-revision: hash
license: spdx
```

The value of a label can be either of `text`, `url`, `semver`, `hash` or `rfc3339`:

Schema	Description
text	Anything
rfc3339	A time, formatted according to RFC 3339
semver	A semantic version
url	A URI as described in RFC 3986
hash	Either a short or a long Git hash
spdx	An SPDX license identifier
email	An email address conforming to RFC 5322

By default, Hadolint ignores any label that is not specified in the label schema. To warn against such additional labels, turn on strict labels, using the command line:

```
hadolint --strict-labels --require-label version:semver Dockerfile
```



or the config file:

```
strict-labels: true
```



When strict labels is enabled, but no label schema is specified, `hadolint` will warn if any label is present.

Note on dealing with variables in labels

It is a common pattern to fill the value of a label not statically, but rather dynamically at build time by using a variable:

```
FROM debian:buster
ARG VERSION="du-jour"
LABEL version="${VERSION}"
```



To allow this, the label schema must specify `text` as value for that label:

```
label-schema:
  version: text
```



Integrations

To get most of `hadolint`, it is useful to integrate it as a check in your CI or into your editor, or as a pre-commit hook, to lint your `Dockerfile` as you write it. See our [Integration](#) docs.

- [Code Review Platform Integrations](#)
- [Continuous Integrations](#)
- [Editor Integrations](#)
- [Version Control Integrations](#)

Rules

An incomplete list of implemented rules. Click on the error code to get more detailed information.

- Rules with the prefix `DL` are from `hadolint`. Have a look at `Rules.hs` to find the implementation of the rules.
- Rules with the `SC` prefix are from **ShellCheck** (only the most common rules are listed, there are dozens more).

Please [create an issue](#) if you have an idea for a good rule.

Rule	Default Severity	Description
DL1001	Ignore	Please refrain from using inline ignore pragmas <code># hadolint ignore=DLxxxx</code> .
DL3000	Error	Use absolute WORKDIR.
DL3001	Info	For some bash commands it makes no sense running them in a Docker container like ssh, vim, shutdown, service, ps, free, top, kill, mount, ifconfig.
DL3002	Warning	Last user should not be root.
DL3003	Warning	Use WORKDIR to switch to a directory.
DL3004	Error	Do not use sudo as it leads to unpredictable behavior. Use a tool like gosu to enforce root.
DL3006	Warning	Always tag the version of an image explicitly.
DL3007	Warning	Using latest is prone to errors if the image will ever update. Pin the version explicitly to a release tag.
DL3008	Warning	Pin versions in <code>apt-get install</code> .
DL3009	Info	Delete the apt-get lists after installing something.
DL3010	Info	Use ADD for extracting archives into an image.
DL3011	Error	Valid UNIX ports range from 0 to 65535.
DL3012	Error	Multiple <code>HEALTHCHECK</code> instructions.
DL3013	Warning	Pin versions in pip.
DL3014	Warning	Use the <code>-y</code> switch.
DL3015	Info	Avoid additional packages by specifying <code>--no-install-recommends</code> .
DL3016	Warning	Pin versions in <code>npm</code> .
DL3018	Warning	Pin versions in <code>apk add</code> . Instead of <code>apk add <package></code> use <code>apk add <package>=<version></code> .

Rule	Default Severity	Description
DL3019	Info	Use the <code>--no-cache</code> switch to avoid the need to use <code>--update</code> and remove <code>/var/cache/apk/*</code> when done installing packages.
DL3020	Error	Use <code>COPY</code> instead of <code>ADD</code> for files and folders.
DL3021	Error	<code>COPY</code> with more than 2 arguments requires the last argument to end with <code>/</code>
DL3022	Warning	<code>COPY --from</code> should reference a previously defined <code>FROM</code> alias
DL3023	Error	<code>COPY --from</code> cannot reference its own <code>FROM</code> alias
DL3024	Error	<code>FROM</code> aliases (stage names) must be unique
DL3025	Warning	Use arguments JSON notation for CMD and ENTRYPOINT arguments
DL3026	Error	Use only an allowed registry in the <code>FROM image</code>
DL3027	Warning	Do not use <code>apt</code> as it is meant to be an end-user tool, use <code>apt-get</code> or <code>apt-cache</code> instead
DL3028	Warning	Pin versions in gem install. Instead of <code>gem install <gem></code> use <code>gem install <gem>:<version></code>
DL3029	Warning	Do not use <code>--platform</code> flag with FROM.
DL3030	Warning	Use the <code>-y</code> switch to avoid manual input <code>yum install -y <package></code>
DL3032	Warning	<code>yum clean all</code> missing after yum command.
DL3033	Warning	Specify version with <code>yum install -y <package>-<version></code>
DL3034	Warning	Non-interactive switch missing from <code>zypper</code> command: <code>zypper install -y</code>
DL3035	Warning	Do not use <code>zypper dist-upgrade</code> .
DL3036	Warning	<code>zypper clean</code> missing after zypper use.
DL3037	Warning	Specify version with <code>zypper install -y <package>[=<version></code> .
DL3038	Warning	Use the <code>-y</code> switch to avoid manual input <code>dnf install -y <package></code>

Rule	Default Severity	Description
DL3040	Warning	<code>dnf clean all</code> missing after <code>dnf</code> command.
DL3041	Warning	Specify version with <code>dnf install -y <package>-<version></code>
DL3042	Warning	Avoid cache directory with <code>pip install --no-cache-dir <package></code> .
DL3043	Error	<code>ONBUILD</code> , <code>FROM</code> or <code>MAINTAINER</code> triggered from within <code>ONBUILD</code> instruction.
DL3044	Error	Do not refer to an environment variable within the same <code>ENV</code> statement where it is defined.
DL3045	Warning	<code>COPY</code> to a relative destination without <code>WORKDIR</code> set.
DL3046	Warning	<code>useradd</code> without flag <code>-l</code> and high UID will result in excessively large Image.
DL3047	Info	<code>wget</code> without flag <code>--progress</code> will result in excessively bloated build logs when downloading larger files.
DL3048	Style	Invalid Label Key
DL3049	Info	Label <code><label></code> is missing.
DL3050	Info	Superfluous label(s) present.
DL3051	Warning	Label <code><label></code> is empty.
DL3052	Warning	Label <code><label></code> is not a valid URL.
DL3053	Warning	Label <code><label></code> is not a valid time format - must conform to RFC3339.
DL3054	Warning	Label <code><label></code> is not a valid SPDX license identifier.
DL3055	Warning	Label <code><label></code> is not a valid git hash.
DL3056	Warning	Label <code><label></code> does not conform to semantic versioning.
DL3057	Ignore	<code>HEALTHCHECK</code> instruction missing.
DL3058	Warning	Label <code><label></code> is not a valid email format - must conform to RFC5322.

Rule	Default Severity	Description
DL3059	Info	Multiple consecutive <code>RUN</code> instructions. Consider consolidation.
DL3060	Info	<code>yarn cache clean</code> missing after <code>yarn install</code> was run.
DL3061	Error	Invalid instruction order. Dockerfile must begin with <code>FROM</code> , <code>ARG</code> or comment.
DL4000	Error	<code>MAINTAINER</code> is deprecated.
DL4001	Warning	Either use <code>Wget</code> or <code>Curl</code> but not both.
DL4003	Warning	Multiple <code>CMD</code> instructions found.
DL4004	Error	Multiple <code>ENTRYPOINT</code> instructions found.
DL4005	Warning	Use <code>SHELL</code> to change the default shell.
DL4006	Warning	Set the <code>SHELL</code> option <code>-o pipefail</code> before <code>RUN</code> with a pipe in it
SC1000		<code>\$</code> is not used specially and should therefore be escaped.
SC1001		This <code>\c</code> will be a regular <code>'c'</code> in this context.
SC1007		Remove space after <code>=</code> if trying to assign a value (or for empty string, use <code>var='' ...</code>).
SC1010		Use semicolon or linefeed before <code>done</code> (or quote to make it literal).
SC1018		This is a unicode non-breaking space. Delete it and retype as space.
SC1035		You need a space here
SC1045		It's not <code>foo &; bar</code> , just <code>foo & bar</code> .
SC1065		Trying to declare parameters? Don't. Use <code>()</code> and refer to params as <code>\$1</code> , <code>\$2</code> etc.
SC1066		Don't use <code>\$</code> on the left side of assignments.
SC1068		Don't put spaces around the <code>=</code> in assignments.

Rule	Default Severity	Description
SC1077		For command expansion, the tick should slant left (` vs `).
SC1078		Did you forget to close this double-quoted string?
SC1079		This is actually an end quote, but due to next char, it looks suspect.
SC1081		Scripts are case sensitive. Use <code>if</code> , not <code>If</code> .
SC1083		This <code>{/}</code> is literal. Check expression (missing <code>;/\n</code> ?) or quote it.
SC1086		Don't use <code>\$</code> on the iterator name in for loops.
SC1087		Braces are required when expanding arrays, as in <code>\${array[idx]}</code> .
SC1095		You need a space or linefeed between the function name and body.
SC1097		Unexpected <code>==</code> . For assignment, use <code>=</code> . For comparison, use <code>[..]</code> or <code>[[..]]</code> .
SC1098		Quote/escape special characters when using <code>eval</code> , e.g. <code>eval "a=(b)"</code> .
SC1099		You need a space before the <code>#</code> .
SC2002		Useless cat. Consider <code>cmd < file ..</code> or <code>cmd file ..</code> instead.
SC2015		Note that <code>A && B C</code> is not if-then-else. C may run when A is true.
SC2026		This word is outside of quotes. Did you intend to 'nest <code>""</code> single quotes <code>""</code> instead'?
SC2028		<code>echo</code> won't expand escape sequences. Consider <code>printf</code> .
SC2035		Use <code>./*glob*</code> or <code>-- *glob*</code> so names with dashes won't become options.
SC2039		In POSIX sh, something is undefined.
SC2046		Quote this to prevent word splitting
SC2086		Double quote to prevent globbing and word splitting.

Rule	Default Severity	Description
SC2140		Word is in the form <code>"A"B"C"</code> (B indicated). Did you mean <code>"ABC"</code> or <code>"A\"B\"C"</code> ?
SC2154		var is referenced but not assigned.
SC2155		Declare and assign separately to avoid masking return values.
SC2164		Use <code>cd ... exit</code> in case <code>cd</code> fails.

Develop

If you are an experienced Haskell, we would be very grateful if you would tear our code apart in a review.

To compile, you will need a recent Haskell environment and `cabal-install`.

Setup

1. Clone repository

```
git clone --recursive git@github.com:hadolint/hadolint.git
```



2. Install dependencies and compile source

```
cabal configure
cabal build
```



3. (Optional) Install Hadolint on your system

```
cabal install
```



REPL

The easiest way to try out the parser is using the REPL.

```
# start the repl
cabal repl
# overload strings to be able to use Text
:set -XOverloadedStrings
# import parser library
import Language.Docker
# parse instruction and look at AST representation
parseText "FROM debian:jessie"
```



Tests

Compile with unit tests and run them:

```
cabal configure --enable-tests
cabal build --enable-tests
cabal test
```



Run integration tests:

```
./integration_test.sh
```



AST

Dockerfile syntax is fully described in the [Dockerfile reference](#). Just take a look at [Syntax.hs](#) in the `language-docker` project to see the AST definition.

Building against custom libraries

Hadolint uses many libraries to do the dirty work. In particular, `language-docker` is used to parse Dockerfiles and produce an AST which then can be analyzed. To build Hadolint against a custom version of such libraries, do the following. This example uses `language-docker`, but it would work with any other library as well.

