Environment Variables

A common way to customize the build process is to define environment variables, which can be accessed from any stage in your build process.

The best way to define an environment variable depends on what type of information it will contain, and when you need to change it:

- if it does *not* contain sensitive information and should be available to forks – add it to your .travis.yml
- if it *does* contain sensitive information, and is the same for all branches – encrypt it and add it to your .travis.yml
- if it *does* contain sensitive information, and might be different for different branches – add it to your Repository Settings

Defining Public Variables in .travis.yml

Public variables defined in `.travis.yml` are tied to a certain commit. Changing them requires a new commit, restarting an old build uses the old values. They are also available automatically on forks of the repository.

Define variables in `.travis.yml` that:

- are needed for the build to run and that don't contain sensitive data. For instance, a test suite for a Ruby application might require `$RACK_ENV` to be set to `test` .
- differ per branch.
- differ per job.

Define environment variables in your `.travis.yml` in the `env` key, quoting special characters such as asterisks (`*`). One build will be triggered for each line in the `env` array.

.travis.yml

YAML

```yaml
env:
  - DB=postgres
  - SH=bash
  - PACKAGE_VERSION="1.0.*"
```

If you define a variable with the same name in `.travis.yml` and in the Repository Settings, the one in `.travis.yml` takes precedence. If you define a variable in `.travis.yml` as both encrypted and unencrypted, the one defined later in the file takes precedence.

Variables' values are passed to the generated build script verbatim. So make sure to escape any Bash special characters accordingly. In particular, if a value contains spaces, you need to put quotes around that value. E.g. `a long phrase` should be written as `"a long phrase"` .

Defining Multiple Variables per Item

If you need to specify several environment variables for each build, put them all on the same line in the `env` array:

.travis.yml

YAML

```yaml
rvm:
  - 1.9.3
  - rbx-3
env:
  - FOO=foo BAR=bar
  - FOO=bar BAR=foo
```

this configuration triggers **4 individual builds**:

1. Ruby 1.9.3 with `FOO=foo` and `BAR=bar`
2. Ruby 1.9.3 with `FOO=bar` and `BAR=foo`
3. Rubinius latest version (rbx-3) with `FOO=foo` and `BAR=bar`
4. Rubinius latest version (rbx-3) with `FOO=bar` and `BAR=foo`

Global Variables

Sometimes you may want to use environment variables that are global to the matrix, i.e. they're inserted into each matrix row. That may include keys, tokens, URIs or other data that is needed for every build. In such cases, instead of manually adding such keys to each `env` line in matrix, you can use `global` and `matrix` keys to differentiate between those two cases. For example:

.travis.yml

```yaml
env:
  global:
    - CAMPFIRE_TOKEN=abc123
    - TIMEOUT=1000
  jobs:
    - USE_NETWORK=true
    - USE_NETWORK=false
```

triggers builds with the following `env` rows:

```bash
USE_NETWORK=true CAMPFIRE_TOKEN=abc123 TIMEOUT=1000
USE_NETWORK=false CAMPFIRE_TOKEN=abc123 TIMEOUT=1000
```

Defining encrypted variables in .travis.yml

Before adding sensitive data such as API credentials to your `.travis.yml` you need to encrypt it. Encrypted variables are not available to untrusted builds such as pull requests coming from another repository.

A `.travis.yml` file containing encrypted variables looks like this:

.travis.yml

```yaml
env:
  global:
    - secure: mcUCykGm4bUZ3CaW6AxrIMFzuAYjA98VIz6YmYTmM0/8sp/B/54JtQS/j0ehCD6B5BwyW6diVcaQA2c7bovI23GyeTT+
  jobs:
    - USE_NETWORK=true
    - USE_NETWORK=false
    - secure: <you can also put encrypted vars inside matrix>
```

Encrypted environment variables are not available to pull requests from forks due to the security risk of exposing such information to unknown code.

If you define a variable with the same name in `.travis.yml` and in the Repository Settings, the one in `.travis.yml` takes precedence. If you define a variable in `.travis.yml` as both encrypted and unencrypted, the one defined later in the file takes precedence.

Encrypting environment variables

Encrypt environment variables with the public key attached to your repository using the `travis` gem:

1. If you do not have the `travis` gem installed, run `gem install travis` (or `brew install travis` on macOS).
2. In your repository directory:
   - If you are using https://travis-ci.com, see Encryption keys – Usage.
   - If you are using https://travis-ci.org, run:

     Bash
     ```
     travis encrypt MY_SECRET_ENV=super_secret --add env.global
     ```
3. Commit the changes to your `.travis.yml` .

Encryption and decryption keys are tied to the repository. If you fork a project and add it to Travis CI, it will not have access to the encrypted variables.

The encryption scheme is explained in more detail in Encryption keys.

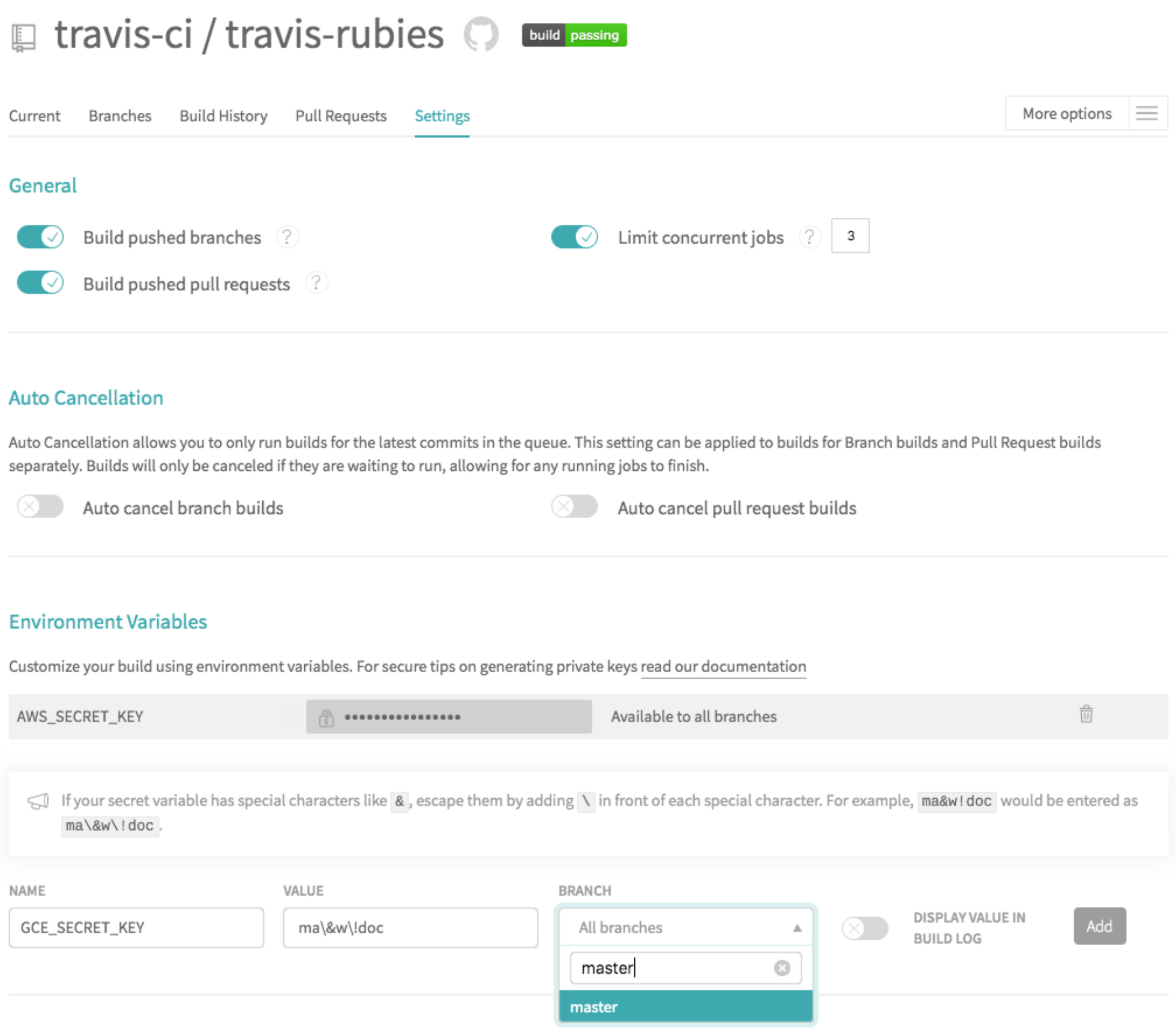
Defining Variables in Repository Settings

Variables defined in repository settings are the same for all builds, and when you restart an old build, it uses the latest values. These variables are not automatically available to forks.

Define variables in the Repository Settings that:

- differ per repository.
- contain sensitive data, such as third-party credentials.

To define variables in Repository Settings, make sure you're logged in, navigate to the repository in question, choose "Settings" from the "More options" menu, and click on "Add new variable" in the "Environment Variables" section. Restrict the environment variable to a specific branch by selecting which branch it should be available to.



General

Build pushed branches ✓

Build pushed pull requests ✓

Limit concurrent jobs ✓ 3

Auto Cancellation

Auto Cancellation allows you to only run builds for the latest commits in the queue. This setting can be applied to builds for Branch builds and Pull Request builds separately. Builds will only be canceled if they are waiting to run, allowing for any running jobs to finish.

Auto cancel branch builds

Auto cancel pull request builds

Environment Variables

Customize your build using environment variables. For secure tips on generating private keys read our documentation

AWS_SECRET_KEY	••••••••••••••••	Available to all branches

If your secret variable has special characters like `&`, escape them by adding `\` in front of each special character. For example, `ma&w!doc` would be entered as `ma\&w\!doc`.

NAME	VALUE	BRANCH	DISPLAY VALUE IN BUILD LOG	
GCE_SECRET_KEY	ma\&w\!doc	All branches / master		Add

Environment Variables in the Repository Settings

See the note above on how to format variables' values correctly.

By default, the value of these new environment variables is hidden from the `export` line in the logs. This corresponds to the behavior of encrypted variables in your `.travis.yml`. The variables are stored encrypted in our systems, and get decrypted when the build script is generated.

Similarly, we do not provide these values to untrusted builds, triggered by pull requests from another repository.

As an alternative to the web interface, you can also use the CLI's `env` command.

If you define a variable with the same name in `.travis.yml` *and in the Repository Settings, the one in* `.travis.yml` *takes precedence.*

Build Config Reference

You can find more information on the build config format for Environment Variables in our Travis CI Build Config Reference.

Convenience Variables

To make using encrypted environment variables easier, the following environment variables are available:

- `TRAVIS_SECURE_ENV_VARS` is set to `true` if you have defined any encrypted variables, including variables defined in the Repository Settings, and `false` if you have not.
- `TRAVIS_PULL_REQUEST` is set to the pull request number if the current job is a pull request build, or `false` if it's not.

Default Environment Variables

The following default environment variables are available to all builds.

- `CI=true`
- `TRAVIS=true`
- `CONTINUOUS_INTEGRATION=true`
- `DEBIAN_FRONTEND=noninteractive`
- `HAS_JOSH_K_SEAL_OF_APPROVAL=true`
- `USER=travis`
- `HOME` is set to `/home/travis` on Linux, `/Users/travis` on MacOS, and `/c/Users/travis` on Windows.
- `LANG=en_US.UTF-8`
- `LC_ALL=en_US.UTF-8`
- `RAILS_ENV=test`
- `RACK_ENV=test`
- `MERB_ENV=test`
- `JRUBY_OPTS="--server -Dcext.enabled=false -Xcompile.invokedynamic=false"`
- `JAVA_HOME` is set to the appropriate value.

Additionally, Travis CI sets environment variables you can use in your build, e.g. to tag the build, or to run post-build deployments.

- `TRAVIS_ALLOW_FAILURE`:
  - set to `true` if the job is allowed to fail.
  - set to `false` if the job is not allowed to fail.
- `TRAVIS_APP_HOST`: The name of the server compiling the build script. This server serves certain helper files (such as `gimme`, `nvm`, `sbt`) from `/files` to avoid external network calls; e.g., `curl -O $TRAVIS_APP_HOST/files/gimme`
- `TRAVIS_BRANCH`:

  - for push builds, or builds not triggered by a pull request, this is the name of the branch.
  - for builds triggered by a pull request this is the name of the branch targeted by the pull request.
  - for builds triggered by a tag, this is the same as the name of the tag (`TRAVIS_TAG`).

> *Note that for tags, git does not store the branch from which a commit was tagged.*

- `TRAVIS_BUILD_DIR` : The absolute path to the directory where the repository being built has been copied on the worker.
- `TRAVIS_BUILD_ID` : The id of the current build that Travis CI uses internally.
- `TRAVIS_BUILD_NUMBER` : The number of the current build (for example, “4”).
- `TRAVIS_BUILD_WEB_URL` : URL to the build log.
- `TRAVIS_COMMIT` : The commit that the current build is testing.
- `TRAVIS_COMMIT_MESSAGE` : The commit subject and body, unwrapped.
- `TRAVIS_COMMIT_RANGE` : The range of commits that were included in the push or pull request. (Note that this is empty for builds triggered by the initial commit of a new branch.)
- `TRAVIS_COMPILER` : Indicates the compiler used by the current job (e.g., `clang` , `gcc`).
- `TRAVIS_DEBUG_MODE` : Set to `true` if the job is running in debug mode
- `TRAVIS_DIST` : Indicates the distribution the current job is running on.
- `TRAVIS_EVENT_TYPE` : Indicates how the build was triggered. One of `push` , `pull_request` , `api` , `cron` .
- `TRAVIS_JOB_ID` : The id of the current job that Travis CI uses internally.
- `TRAVIS_JOB_NAME` : The job name if it was specified, or `""` .
- `TRAVIS_JOB_NUMBER` : The number of the current job (for example, “4.1”).
- `TRAVIS_JOB_WEB_URL` : URL to the job log.
- `TRAVIS_OS_NAME` : On multi-OS builds, this value indicates the platform the job is running on. Values are currently `linux` , `osx` and `windows` (beta), to be extended in the future.
- `TRAVIS_CPU_ARCH` : On multi-arch builds, this value indicates the CPU architecture the job is running on. Values are currently `amd64` , `arm64` , `ppc64le` and `s390x` .
- `TRAVIS_OSX_IMAGE` : The `osx_image` value configured in `.travis.yml` . If this is not set in `.travis.yml` , it is empty.
- `TRAVIS_PULL_REQUEST` : The pull request number if the current job is a pull request, “false” if it’s not a pull request.
- `TRAVIS_PULL_REQUEST_BRANCH` :
  - if the current job is a pull request, the name of the branch from which the PR originated.
  - if the current job is a push build, this variable is empty (`""`).
- `TRAVIS_PULL_REQUEST_SHA` :
  - if the current job is a pull request, the commit SHA of the HEAD commit of the PR.
  - if the current job is a push build, this variable is empty (`""`).
- `TRAVIS_PULL_REQUEST_SLUG` :
  - if the current job is a pull request, the slug (in the form `owner_name/repo_name`) of the repository from which the PR originated.
  - if the current job is a push build, this variable is empty (`""`).
- `TRAVIS_REPO_SLUG` : The slug (in form: `owner_name/repo_name`) of the repository currently being built.

- `TRAVIS_SECURE_ENV_VARS` :
  - set to `true` if there are any encrypted environment variables.
  - set to `false` if no encrypted environment variables are available.
- `TRAVIS_SUDO` : `true` or `false` based on whether `sudo` is enabled.
- `TRAVIS_TEST_RESULT` : **0** if all commands in the `script` section (up to the point this environment variable is referenced) have exited with zero; **1** otherwise.
- `TRAVIS_TAG` : If the current build is for a git tag, this variable is set to the tag's name, otherwise it is empty (`""`).
- `TRAVIS_BUILD_STAGE_NAME` : The build stage in capitalized form, e.g. `Test` or `Deploy` . If a build does not use build stages, this variable is empty (`""`).

Language-specific builds expose additional environment variables representing the current version being used to run the build. Whether or not they're set depends on the language you're using.

- `TRAVIS_DART_VERSION`
- `TRAVIS_GO_VERSION`
- `TRAVIS_HAXE_VERSION`
- `TRAVIS_JDK_VERSION`
- `TRAVIS_JULIA_VERSION`
- `TRAVIS_NODE_VERSION`
- `TRAVIS_OTP_RELEASE`
- `TRAVIS_PERL_VERSION`
- `TRAVIS_PHP_VERSION`
- `TRAVIS_PYTHON_VERSION`
- `TRAVIS_R_VERSION`
- `TRAVIS_RUBY_VERSION`
- `TRAVIS_RUST_VERSION`
- `TRAVIS_SCALA_VERSION`

Other software specific environment variables are set when the software or service is installed or started, and contain the version number:

- `TRAVIS_MARIADB_VERSION`

The following environment variables are available for Objective-C builds.

- `TRAVIS_XCODE_SDK`
- `TRAVIS_XCODE_SCHEME`
- `TRAVIS_XCODE_PROJECT`
- `TRAVIS_XCODE_WORKSPACE`

Travis CI