Travis CI Cheat Sheet

A comprehensive cheat sheet covering essential Travis CI configurations, commands, and best practices for continuous integration and deployment.

Core Concepts & Configuration

Basic `.travis.yml` Structure

A basic `.travis.yml` file specifies the language and build steps:

```yaml
language: ruby
rvm:
  - 2.7
before_install:
  - gem install bundler
install:
  - bundle install
script:
  - bundle exec rspec
```

Key components:

- `language`: Specifies the programming language.
- `rvm` (or equivalent): Specifies the version manager and versions.
- `before_install`: Commands to run before installing dependencies.
- `install`: Commands to install dependencies.
- `script`: Commands to run the build/test suite.

Language Support

`language: ruby`	Specifies the Ruby language environment.
`language: node_js`	Specifies the Node.js environment.
`language: python`	Specifies the Python environment.
`language: java`	Specifies the Java environment.

Build Lifecycle Stages

Travis CI build lifecycle consists of distinct stages:

- `before_install`: Prepare the environment.
- `install`: Install dependencies.
- `before_script`: Run commands before the main script.
- `script`: Run the primary build script (tests, etc.).
- `after_success` / `after_failure`: Commands to run based on the script's success.
- `after_script`: Always runs regardless of build outcome.
- `before_deploy`: Run before deploying code.
- `deploy`: Deploy the code to a provider.
- `after_deploy`: Run commands after successful deployment.

Advanced Configuration

Environment Variables

`env:`	Define environment variables for the build. Can be global or matrix-specific.
	`env:`
	`  global:`
	`  - secure: ENCRYPTED_PASSWORD`
	`  matrix:`
	`  - TEST_SUITE=unit`
	`  - TEST_SUITE=integration`

Secure Variables	Sensitive data should be encrypted using the Travis CI CLI and stored as `secure:` variables.

Build Matrix

A build matrix allows you to test your code against multiple configurations.

```yaml
matrix:
  include:
    - rvm: 2.6
      gemfile: gemfiles/rails-5.2.gemfile
    - rvm: 2.7
      gemfile: gemfiles/rails-6.0.gemfile
```

You can exclude specific configurations:

```yaml
matrix:
  exclude:
    - rvm: 2.5
      gemfile: gemfiles/rails-6.0.gemfile
```

Caching Dependencies

`cache:`	Enable caching to speed up builds by reusing dependencies.
	`cache:`
	`  directories:`
	`  - node_modules`
	`  - vendor/bundle`

Common directories	`node_modules`, `vendor/bundle`, and other dependency directories can be cached.

Deployment

Basic Deployment Configuration

Travis CI supports deployment to various providers. Here's an example for deploying to Heroku:

```yaml
deploy:
  provider: heroku
  api_key:
    secure: ENCRYPTED_HEROKU_API_KEY
  app: your-heroku-app-name
  on:
    branch: master
```

Key components:

- `provider`: Specifies the deployment provider.
- `api_key`: Your API key for the provider (encrypted).
- `app`: The name of your application on the provider.
- `on`: Conditions for deployment (e.g., branch).

Conditional Deployment

`on: branch: master`	Deploys only when the build is triggered from the `master` branch.
`on: tags: true`	Deploys only when a tagged commit is built.

Deployment Providers

Travis CI supports a wide range of deployment providers, including:

- Heroku
- AWS (S3, Elastic Beanstalk)
- Firebase
- GitHub Pages
- PyPI
- npm
- and many more.

Tips and Tricks

Debugging Travis CI Builds

Debugging failed builds:

- Check the Travis CI build logs for error messages.
- Enable debug mode by setting `travis_debug: true` in your `.travis.yml`.
- Use SSH access for interactive debugging (requires a paid plan).
- Add `echo` statements in your `.travis.yml` to print variable values and execution flow.

Optimizing Build Times

Caching	Cache dependencies to reduce installation time.
Parallelization	Run tests in parallel using tools like `parallel_test` (for Ruby) or `tox` (for Python).
Selective Testing	Run only the necessary tests based on changed files.

Common Issues and Solutions

Common issues:

- Incorrect `rvm` or language version.
- Missing dependencies.
- Test failures due to environment differences.
- Deployment failures due to incorrect credentials.

Solutions:

- Double-check your `.travis.yml` configuration.
- Ensure all dependencies are listed in your dependency management file (e.g., `Gemfile`, `package.json`).
- Use environment variables to handle sensitive data.
- Test your deployment process locally before pushing to Travis CI.