

Tachikoma next

sanemat {AT} tachikoma.io

- Write shell script or ruby script. tachikoma gem is going to dead.
- We got ready for essential tools.
- tachikoma gem はしにつつあるので shell script か ruby script で書け
- 必要なツールは用意した

Bundle update regularly, frequently

2013-05-31 - Kensuke Nagae @kyanny

[Continuous gem dependency updating with Jenkins and Pull Request](#)

2013-06-23 - Kenichi Murahashi @sanemat

[Tachikoma gem first version v2.0.0 \(not published\) commit](#)

2014-09-17 - Kenichi Murahashi @sanemat

[Release announce about tachikoma.io](#)

Motivation

- Writing shell script is hard
- Many products / projects have almost same script
- shell script で書くのがそもそもツライ
- ほぼ同ジスクリプトがプロジェクトごとに散らばるのでツライ

Tachikoma gem

pros

- write yaml, we can run command line with gems.
- yaml の設定を置くと、gem とコマンドラインで完結する

cons

- Not customizable enough
- Build your own execute environment
- First I image that I run this in own jenkins
 - CI as a service advances rapidly
- カスタマイズ性低い
- 実行環境は自分で作成する必要がある
- 設計思想として、自前 jenkins で動かすのを第一に考えていた
 - CI as a service の発展

tachikoma.io

pros

- Easy to use with GitHub OAuth, and toggle from webUI
- Configure by .tachikoma.yml
- GitHub OAuth で連携, webUI からtoggle 一発
- .tachikoma.yml で設定

cons

- Too hard to get user's permission about reading/writing on public/private repo
- In public version, user can choose permitting “public write”, only I use this feature (user count is over 300)
- Choose “public write” is very useful because I can stack my commit above bot commit.
- Not customizable enough
- リポジトリの読み書き権限もらうのが非常に苦戦
- public 版も、ユーザー権限で push 出来るようにするものはあるが、300+ のユーザー中利用者 1(自分のみ)
- ユーザー権限で push できると、bot のcommit の上に自分のコミット 詰めるので、すごい便利なのだが...
- カスタマイズ性低い

Swinging back

揺り戻し

- shell script and ruby script is enough
- shell script や ruby script で書けばいいじゃん

But

しかし

Still hard

やっぱりつらい

Why? I know I can use Octokit.rb and use GitHub API. But...

shell script or Ruby script で書くの何がツライのか直に Octokit 使えばいいだけなのわかるけど、いちいち API 調べてとかダルい

Improve usability, concentrate on issue

対象を絞ってより使いやすくするアプローチ

Ruby gem case

- [deppbot](#) webservice
- [circleci-bundle-update-pr](#) gem
- [ci-bundle-update](#) gem

Node.js npm case

- [greenkeeper](#) webservice

Opinion & Experience

ここから知見 + 私見 + 主張

“I want to customize!” appears infinitely, depends on qualities of project and product.

カスタマイズ欲求がプロジェクト・プロダクトの特性によって無限に出てくる (はず)

- Use mongodb
- Use mecab (additional libraries)
- Only business day morning
- No Ruby or Rails project on top level on project
- Multi Ruby projects
- Use npm with same way
- mongodb 使いたい
- mecab 使いたい
- 頻度は平日の朝に来て欲しい
- プロジェクト構成が、トップレベルに Ruby や Rails 来てない
- 複数 Ruby プロジェクトが入っている
- Node.js npm もやりたい
- This is re-invent Travis-ci and CircleCI!?

これは、travis-ci, CircleCI を再発明することになるのでは??

Move to application of CI native

CI native のアプリケーションへ

Run on CI is enough.

CI 上で動けばいいじゃん

Going on ahead; Is “bundle update as a service” unnecessary?

先回り では、bundle update as a service は不要?

So tachikoma.io and deppbot are unnecessary? Are you shut down tachikoma.io?

では、tachikoma.io や deppbot は不要なのか? tachikoma.io 止めちゃうの?

It is too hard to write scripts at all. The setting is difficult, even me.
tachikoma.io is useful at all :)

やっぱり、スクリプト書くのめんどい... 設定メンドイな...わかってる自分でやるのもメンドイぐらいなので、他の人はたぶんできないなやっぱtachikoma.io 便利だな

I need tachikoma.io !

自分には必要!

Especially for small modules like gems and npms.

とくに、gem や npm のような小さいモジュールには。

a conclusion

結論

Kick `bin/bundle-update.sh` or `bundle-udpate/bin/bundle-update.rb` by [Cron for GitHub](#) or [AWS Lambda Scheduled Event](#).

`bin/bundle-update.sh` あるいは `bin/bundle-update.rb` を [Cron for GitHub](#) あるいは [AWS Lambda Scheduled Event](#) で kick する。

`bin/bundle-update.sh`

```
#!/usr/bin/env bash
set -ev

# only sunday
if [[ -n "${TRAVIS_PULL_REQUEST}" && "${TRAVIS_PULL_REQUEST}" == "false" && "${TRAVIS_BRANCH}" == "master" ]]; then
    # gem prepare
    gem install --no-document git_httpsable-push pull_request-create

    # git prepare
    git config user.name sanemat
    git config user.email foo@example.com
    HEAD_DATE=$(date +%Y%m%d_%H-%M-%S)
    HEAD="tachikoma/update-${HEAD_DATE}"

    # checkout (for TravisCI)
    git checkout -b "${HEAD}" "${TRAVIS_BRANCH}"
```

```

# bundle install
bundle --no-deployment --without nothing --jobs 4

# bundle update
bundle update

git add Gemfile.lock
git commit -m "Bundle update ${HEAD_DATE}"

# git push
git httpsable-push origin "${HEAD}"

# pull request
pull-request-create
fi

exit 0

```

bin/bundle-update.rb あとでかく
bundle-udpate/Gemfile

bundle-udpate/bin/bundle-update.rb

cd bundle-update && bundle exec bin/bundle-update.rb && cd ..

Example [sanemat/ruby-example-rails-banana](#)

We got ready for essential tools 細かいのいろいろ作ったので検討してくれ

- [compare_linker_wrapper](#)
- [saddler](#)
- [env_branch](#)

- [parse_gemspec-cli](#)
- [parse_gemspec](#)
- [github_status_notifier](#)
- [env_pull_request](#)
- [restore_bundled_with](#)
- [git_httpsable-push](#)
- [pull_request-create](#)
- [cron_for_github](#)
- [git_diff_parser](#)

[compare_linker](#)

最近の update as a service ならだいたい持ってる、github 上の diff にリンク貼ったコメントを付けてくれる gem

web サービスなら同時にやってくれる方がいいかもだけど、ライブラリなら tachikoma に内蔵するより、別コマンドの方が良い あと、github api 依存の部分を外した compare-linker-wrapper と bin/compare-linker.sh。あと saddler も使っている。compare-linker-wrapper で出力して、saddler で pull request のコメントに載せている。

```
#!/usr/bin/env bash
set -ev

if [[ -n "${TRAVIS_PULL_REQUEST}" && "${TRAVIS_PULL_REQUEST}" != "false" ]]; then
  # gem prepare
  gem install --no-document saddler saddler-reporter-github \
  compare_linker_wrapper text_to_checkstyle github_status_notifier

  github-status-notifier notify --state pending --context saddler/compare_linker

  git diff --name-only origin/master \
  | grep ".*[gG]emfile.lock$" || RETURN_CODE=$?

  case "$RETURN_CODE" in
    "" ) echo "found" ;;
    "1" )
      echo "not found"
      github-status-notifier notify --state success --context saddler/compare_linker
```

```

        exit 0 ;;
    * )
        echo "Error"
        github-status-notifier notify --state failure --context saddler/compare_linker
        exit $RETURN_CODE ;;
    esac

    git diff --name-only origin/master \
    | grep ".*[gG]emfile.lock$" \
    | xargs compare-linker-wrapper --base origin/master \
        --formatter CompareLinker::Formatter::Markdown \
    | text-to-checkstyle \
    | saddler report \
        --require saddler/reporter/github \
        --reporter Saddler::Reporter::Github::PullRequestComment

    github-status-notifier notify --state success --context saddler/compare_linker
fi

exit 0

```

ruby 版 あとでかく

アプリの依存に載せたくないばあいも、`bundle-update/Gemfile` や `compare-linker/Gemfile` にしておけばいいんじゃないですかね (試してない) なお、そうするとこっちの依存部分についても、定期的に bundle update できる。(ように自分で書けば良い)

npm-check-updates

tachikoma はdavid 使ってるけど、npm-check-updatesの方が筋がいい。最近微妙だけど... bower-update とりこんじったり。+1 と-1 で荒れて、取り外されてめでたし。

課題

gem install がコケて定期実行に失敗することがあるらしい `travis_retry gem install xxx` とすればよいのでは (travis なら)

tachikoma gem は発展的解消できたなでも、コレ設定メンドイな...わかってる自分でやるのもメンドイぐらいなので、他の人はできないな やっぱ tachikoma.io 便利だな

余談

パイプ最高期

shell script なら言語中立にいけるやん! パイプ最高や! 期にノッて書いたライブラリ群

あれ、shell script って環境依存激しいし微妙に方言が有る windows? なにそれうまいの? windows ユーザー多いし、どうせなら多くの人に使ってもらいたい。

saddler toolchain

このツールチェーンでは、C extension を使わない。nokogiri, rugged を避ける標準モジュールを使う nokogiri ではなく rexml ビルドするの大変になる & JRuby などでもいけるように rugged 使わなかったけど、それでは windows で動かないので。。。

golang やりたい (やってない) 期

golang で書き換えたいから golang の勉強はじめようかな うーん n 度目の golang やりたい期 golang だとインストールが curl とかになるのでそれはそれでどうなのよ

おわり