Dealing with large jobs on Travis-CI

W. Adam Koszek

Koszek ORG

wojciech@koszek.com

2022-10-05T17:47:15Z

I'm releasing all my projects through continuous integration, so I end up working with Travis-CI a lot. Travis provides a corresponding diagnostic page for each project I have linked to it from GitHub. For example one of my GitHub projects is here and its Travis-CI subpage would be here. If you look at the link format, you'll understand what I mean. In there you can see what the output of your job was. Most of the jobs are fairly simple and finish within short period of time. For these jobs debugging the build steps is easy: just look at the console output and see what's wrong. It's what I do 95% of time. Below I give hints on how to handle 5% of other cases.

Bigger projects

While working on Sensorama for iOS I had to do something different, since it generates a lot of output from many tools. It has many dependencies on Ruby Gems and CocoaPods, which make it long-running as well. Unless you're careful, you'll make your life and Travis job debugging harder. For me debugging OSX/iOS projects is especially challenging, since it's a flow of: bug, commit, verify cycles (for Linux/UNIX projects you can run Travis-CI environment in a Docker container).

Travis-CI limits for Open Source projects

Travis-CI is entirely free for Open Source, which I find great. Even though its run-time limits are pretty generous, they do exist:

- time limit on jobs: which currently is set to 30 minutes.
- 10 minute watchdog: if your job doesn't output anything in this period, it will be killed.
- size of the output files is limited to 4MB. If you run over it, your job will be killed.

How does the issue look like?

Example (from here):

```
+scan --workspace Sensorama.xcworkspace --scheme SensoramaTests
[08:12:08]: xcrun xcodebuild -list -workspace Sensorama.xcworkspace
No output has been received in the last 10m0s, this potentially indicates a
stalled build or something wrong with the build itself.
The build has been terminated
```

Another one (from here):

```
/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs
The log length has exceeded the limit of 4 MB (this usually means that the test suite is ra
The job has been terminated
```

How to deal with this stuff?

Prettifiers?

For XCode projects just pipe everything what comes from `xcodebuild` through `xcpretty`. If you're using fastlane (and if not, you should), then the `xcpretty` is used automatically. Normally I don't like the tools which obfuscate real command's output, but in this case I had no choice. Also `xcpretty` makes the output understandable.

Output folding

Travis-CI has this cool undocumented feature called "output folding". You can group lines of output which belong to the same group of commands. You'll get them folded together and you don't have to scroll through all the output thanks to it.

How to do it?

First of all, your config file `.travis.yml` for the `script` entry has to call 1 script. So for Sensorama I have:

```
language: objective-c
osx_image: xcode7.3
cache: cocoapods
rvm:
- 2.2
podfile: Sensorama/Podfile
script:
- ./scripts/travis_script.sh
#- ./scripts/script_with_folds
addons:
  ssh_known_hosts:
  - gitlab.com
```

If you want to give it a try, use `script_with_folds`, borrowed from here.

Then it's pretty easy:

```
#!/bin/bash

travis_fold() {
  local action=$1
  local name=$2
  echo -en "travis_fold:${action}:${name}\r"
}

travis_fold start foo

echo "This line appears in the fold's 'header'"

echo "Stuff inside"

sleep 2

echo "More stuff"

travis_fold end foo
```

So every fold has to start with `travis_fold:ACTION:name`, where `ACTION` is either `start` or `end`, and the `name` is whatever you want.

Doing folds easily

For Sensorama I plan to use output folding a lot, so I've devised a simpler way to deal with this stuff:

```
TMP=/tmp/.travis_fold_name

# This is meant to be run from top-level dir. of sensorama-ios

travis_fold() {
  local action=$1
  local name=$2
  echo -en "travis_fold:${action}:${name}\r"
}

travis_fold_start() {
  travis_fold start $1
  echo $1
  /bin/echo -n $1 > $TMP
}

travis_fold_end() {
  travis_fold end `cat ${TMP}`
}
```

This is a more elaborate version of what I showed before, but it lets you to do:

```
(
  travis_fold_start BOOSTRAPPING
  ./build.sh bootstrap
  travis_fold_end
)
```

Without worrying whether the start and end tags for the fold are matched. One of the stupid mistakes I've made in one of the commits is that `action` and `name` mismatched, and then folding is broken.

Summary

For optimizing output logging I'd start from trying to limit job output. In cases where you can't do it, try to use some output "compressors" and prettyfiers. Remember that these things add complexity and can sometimes make debugging harder, as they essentially obfuscate the output of original commands. At the end use output folding on Travis-CI.

Was this article useful? Let me know