Buildtime Trend : What's trending in your build process

What, why and how

Dieter Adriaenssens

Buildtime Trend founder & developer - @dcadriaenssens

NewLine 0x05 - Ghent
April 5th, 2015

Overview

- What is Buildtime Trend
- Why and how
- Demo
- Lessons learned

What is Buildtime Trend?

Buildtime Trend is an Open Source application that uses (timing) data to visualise trends in a build process.

It started with an itch

I was working on a project that was built on Travis CI:

- some builds took longer than others
- no timing information was present in the logs
- which stage took longer?

It started with an itch

I was working on a project that was built on Travis CI:

- some builds took longer than others
- no timing information was present in the logs
- which stage took longer?

Solution: create a script to generate timestamps

Next step : calculate duration and generate chart

Requirements :

- parse the generated timestamps
- calculate duration of the stages
- store the timing data of every build
- generate the chart

Next step : calculate duration and generate chart

Requirements :

- parse the generated timestamps
- calculate duration of the stages
- store the timing data of every build
- generate the chart

One solution : Bash + CSV + gnuplot

Next step : calculate duration and generate chart

Requirements :

- parse the generated timestamps
- calculate duration of the stages
- store the timing data of every build
- generate the chart

One solution : Bash + CSV + gnuplot
Another solution : Python + XML + matplotlib

Problem

I didn't use Python before.

Problem

I didn't use Python before.
A good opportunity to learn Python!

Learning Python

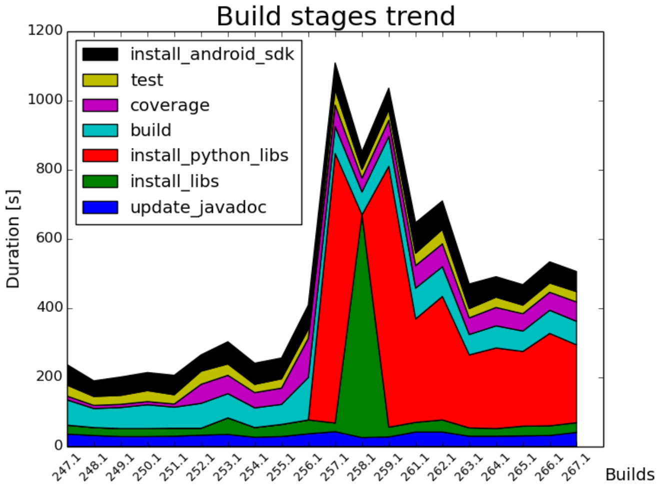
- start with a tutorial
- read documentation
- ask Google and Stack Overflow
- read code from other projects
- get advice from friends and colleagues
- go to talks and conferences

Other helpful things

- check coding style
  - commandline : pep8, pylint
  - online tools : Landscape.io, Scrutinizer, Codacy
- unit testing and test driven development (TDD)
- code coverage
- automate this with Continuous Integration (CI) : Travis CI
- version control : Git, GitHub, ...

Proof of concept

Collection of Bash and Python scripts, generating, analysing and visualising timing data:



Limitations

- scalability of XML
- querying data is limited
- developing new charts is not efficient

Integration with Travis CI

Call Buildtime Trend scripts in .travis.yml file

- generate timestamps
- install dependencies
- analyze timestamps and generate chart
- store timing data (XML) and chart in gh-pages

Integration with Travis CI

This is not ideal!

- scripts become part of the build process
- build duration is influenced
- complicated to set up

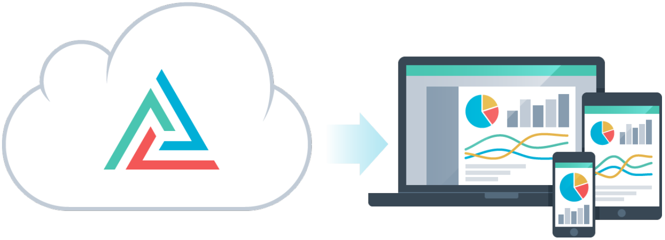
Integration with Travis CI

Possible solutions

- create a service that analyses the logfile after the build is finished
- store the data in a database outside Travis
- host the generated chart(s) outside Travis

But how?

Keen.io



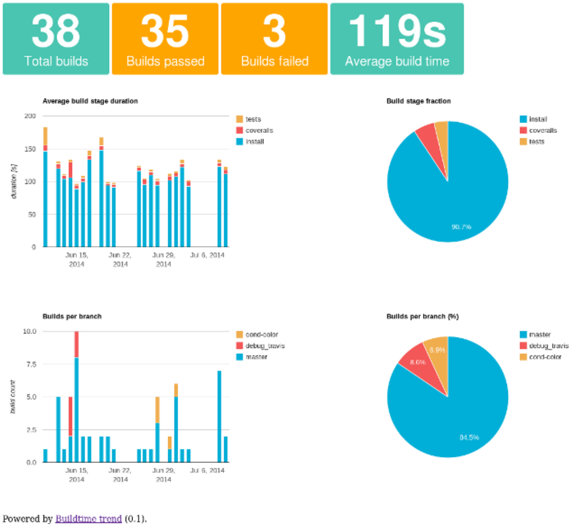
Keen IO's powerful APIs do the heavy lifting for you, so you can gather all the data you want and start getting the answers you need.

Keen.io

- gather data
- store it
- analyse
- visualise
- support for several languages, including Python

Great : this solves the data storage and chart generation problem!

First release - dashboard



38 Total builds | 35 Builds passed | 3 Builds failed | 119s Average build time

Powered by Buildtime trend (0.1).

First release - features

- scripts tightly coupled with Travis CI build
- storing data in XML and chart generation with matplotlib (native mode)
- storing data and chart generation with Keen.io
- chart dashboard deployed to gh-pages

Yeah, it works!

First release - features

- scripts tightly coupled with Travis CI build
- storing data in XML and chart generation with matplotlib (native mode)
- storing data and chart generation with Keen.io
- chart dashboard deployed to gh-pages

Yeah, it works!
But there is still room for improvement.

Travis CI timing data

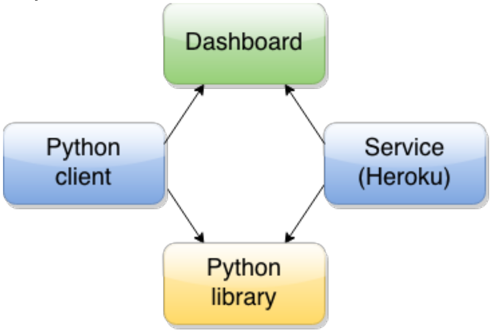
Good news!
Travis CI announces adding timing data to their logs!

Create a service

- get Travis CI build logfile
- parse the timing data
- store data in Keen.io

Rearrange project structure

Create organisation on GitHub to host the different repositories :



Challenges

- let the subprojects work together
- Open Source license
- hosting the service
- triggering the service at the end of a build
- business model

Combine subprojects

- Client and service use dashboard : git submodule
- Client and service use library :
  - create Python package *buildtimetrend*
  - available on *pypi*
  - install as dependency

Open Source license

Transition from GNU General Public License (GPL) v3 to Affero GPLv3

Hosting the service

- CherryPy turns scripts into webservice
    - dashboard
    - badge
    - parse buildlog
- service hosted on Heroku : https://buildtimetrend.herokuapp.com/
- deploy to Heroku with Travis CI build

Deploy to Heroku during Travis CI build

In *.travis.yml* :

Example

```
deploy:
  provider: heroku
  api_key:
    secure: <secure key>
  app:
    deploy-prod: buildtimetrend
```

Branch *deploy-prod* is deployed to app
https://buildtimetrend.herokuapp.com/

Travis CI triggers service

Trigger the service at the end of a Travis CI build in *.travis.yml* :

Example

```
notifications:
  webhooks:
    - https://buildtimetrend.herokuapp.com/travis
```

Travis CI triggers service

- a JSON payload with build data is sent
- service identifies repo and build ID
- build job data and logfiles are retrieved from Travis CI
- build job logfiles are parsed looking for timing tags
- build stages duration is calculated
- build job data is sent to Keen.io

Business model

- offering the service costs money
  - hosting the service
  - storing the data
- Github inspired business model : free for Open Source, paying for private repos
- Keen.io offered to host data for free for Open Source projects. **Thanks, guys!**
- Further development with real data gathering : better feedback

Future development

- caching for badges and dashboard charts
- service : seperate worker for build job processing
- support private Github repos
- support other CI platforms (Jenkins, ...)
- more and improved metrics and trends

Contributions welcome

- use and test the service
- report bugs
- suggest improvements
- clone the project and send pull requests
  - implement a feature
  - fix a bug
  - add a new chart
  - improve dashboard layout
  - ...

Demo

- Service : https://buildtimetrend.herokuapp.com/
- Dashboard : https://buildtimetrend.herokuapp.com/dashboard/buildtimetrend/ lib
- Badges : https://github.com/buildtimetrend/service#badge-examples

Lessons learned

- start simple
- (re)use as much existing tools as possible
- start with what you know, learn new skills as you progress
- try, even if it is all scary at first

Acknowledgements

Big thanks to

- Josh (@dzello) and the other nice people of Keen.io, for their invaluable support!
- @thomasbonte for introducing me to Lean Startup principles
- Alex for telling me about CherryPy
- the Open Source projects testdriving the service
- all the services that power the project
- my friends and family

Questions

Thanks for your attention!
Questions?

Dieter Adriaenssens - @dcadriaenssens

Presentation available on
https://ruleant.github.io/presentations/

Buildtime Trend

- Website : https://buildtimetrend.github.io/
- Service : https://buildtimetrend.herokuapp.com/
- Twitter : @buildtime_trend