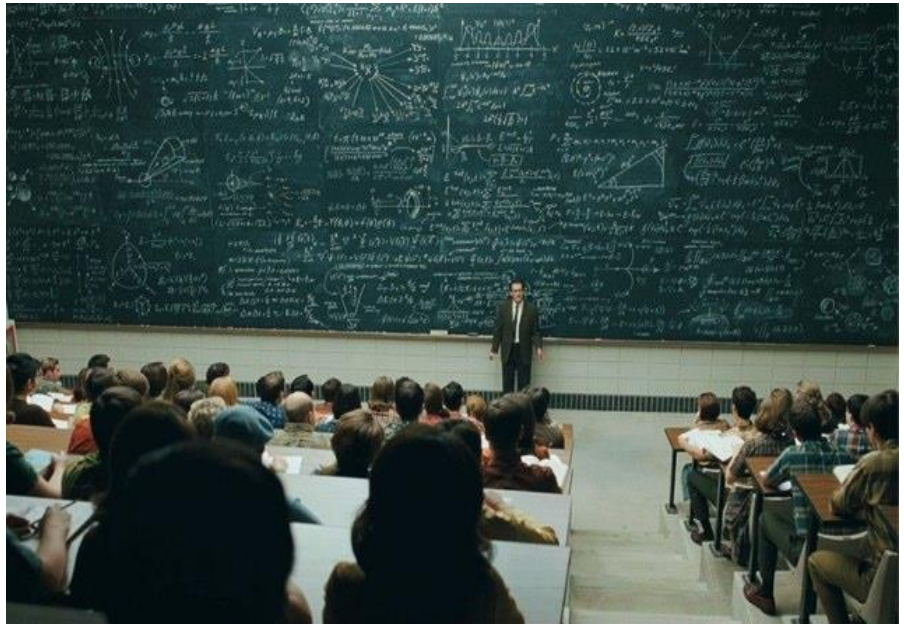
Modern cloud-based workflows for web developers

Focus on quality

SophiaConf 2014

Pawel Kozlowski
@pkozlowski_os
Sophia-Antipolis
03-07-2014

What is the whole talk about?



- Web applications are hard to develop and test
- Open Source community figured out how to make the whole process smooth
- We can get inspired by what works in the Open Source community
- Don’t worry if you doze off - links to the relevant tools are included

1 The problem – it needs to work!

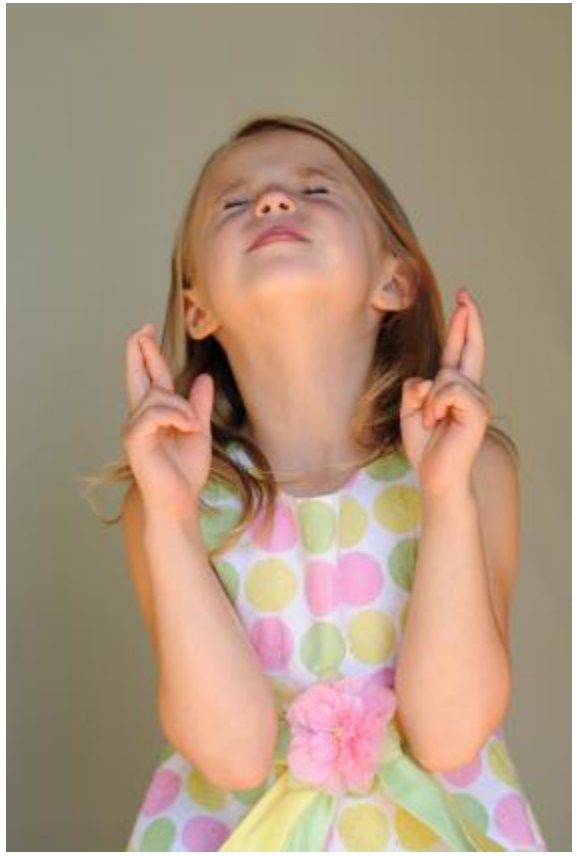
You can choose one of those cars for the same price

Ah, but the engine of the left one breaks down every 100 km. Unexpectedly.

Morale: the thing needs to actually work!

How do we make sure that engines actually work?

Crossing fingers? Like really hard?



We need some serious testing!

Desired test properties:

- Fast and unambiguous feedback
- Automated and cost-effective
- Relevant to real-life scenarios

2

Web application development is (a bit) like building engines

Remember, the thing needs to actually work

And work well on many different platforms

Yes, it is used in many different contexts!

3

Nuts and bolts

Learning from the Open Source community

Linus's Law

"Given enough eyeballs, all bugs are shallow"

"Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix will be obvious to someone."

Central and accessible source-code repository

https://github.com/

Code review

Tooling support is essential for effective code reviews

- Code change tracking (when to review?)
- Clear diffs for any code change (what to review?)
- Ability to comment on any code change (what to fix?)
- History of comment / code changes (was my remark taken into account?)

https://github.com/ariatemplates/hashspace/pull/80

Automatic code review / static analysis

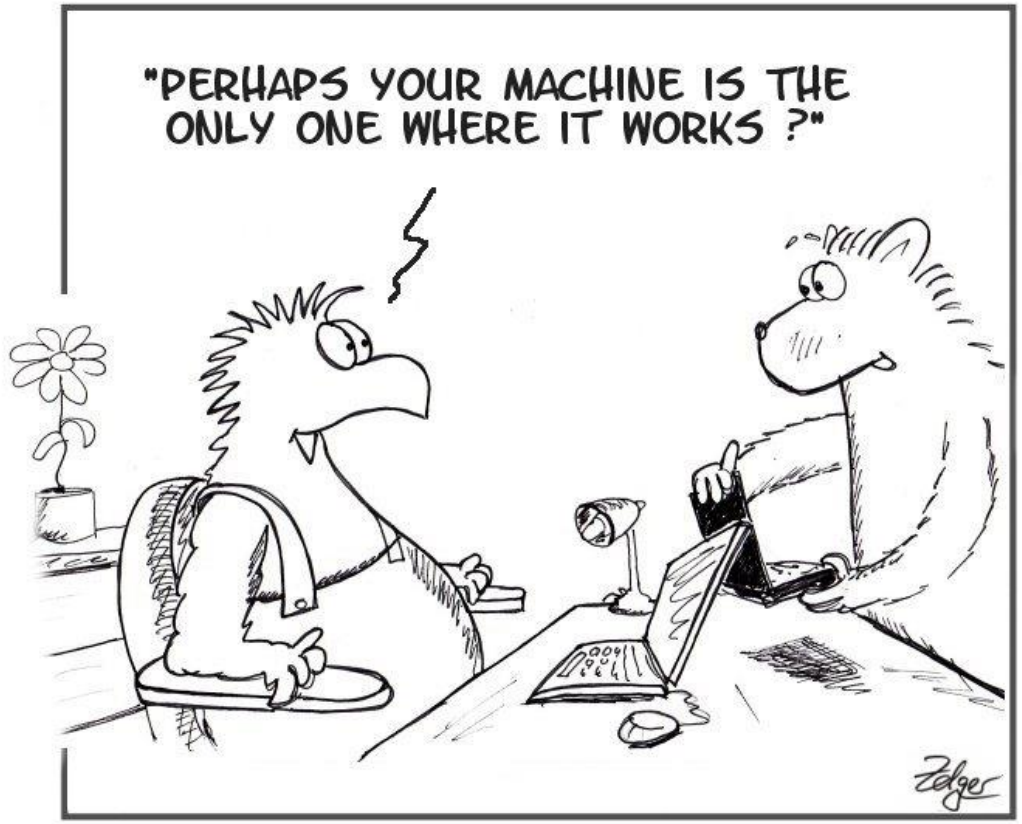
JavaScript is an interpreted language

- Automate any checks that can be automated
  - http://www.jshint.com/
  - https://github.com/mdevils/node-jscs

- Fail build if any of the rules gets violated – no negotiations, no bribing

It works on my machine

Or why we need a CI server



It works on my machine

Meet Travis CI

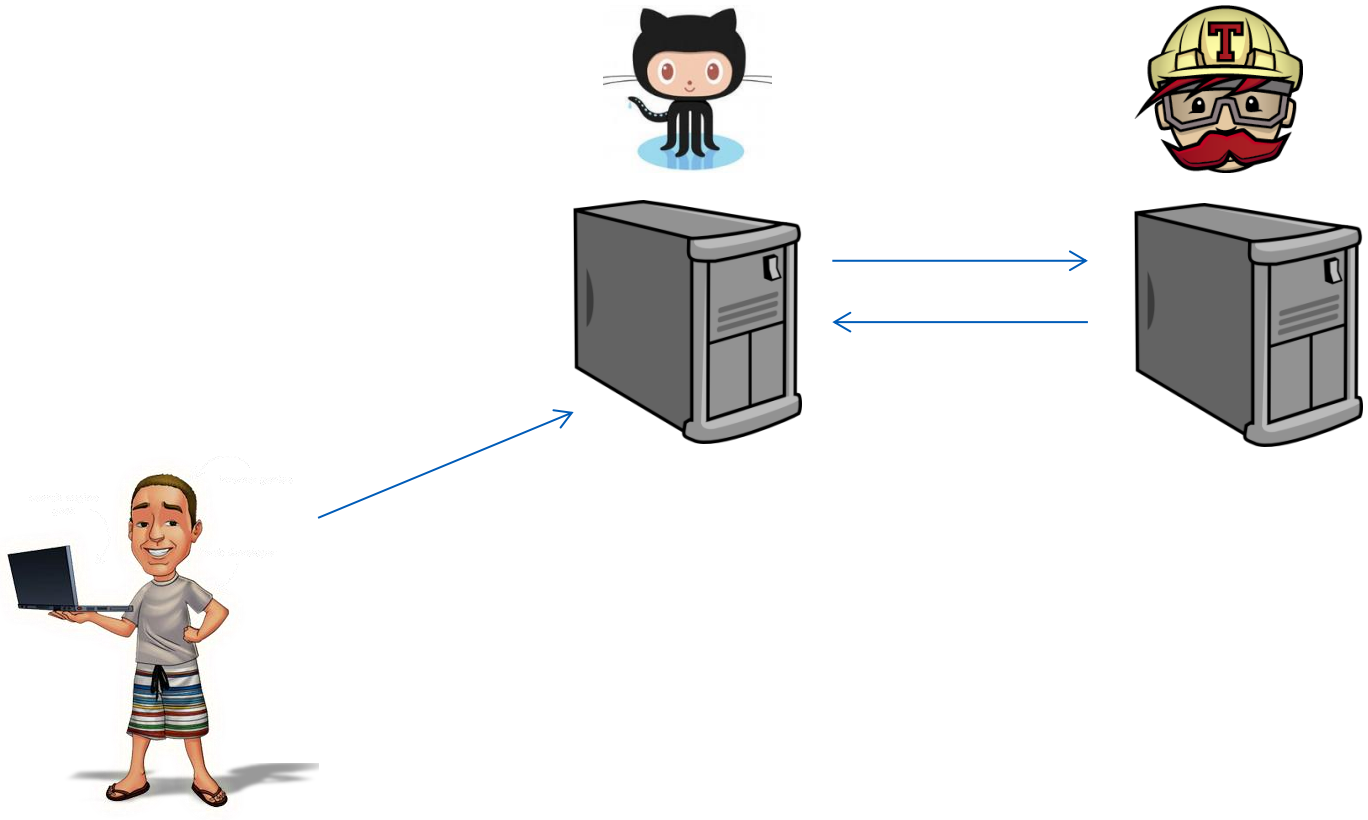
- Gets notified on each code change (main repository and pull requests)
- Starts a build in a completely fresh virtual machine
- Reports results back to GitHub (comments in PRs, badges etc.)

Static analysis is not enough

- Automated unit and component tests are the well established practice in many languages / environments

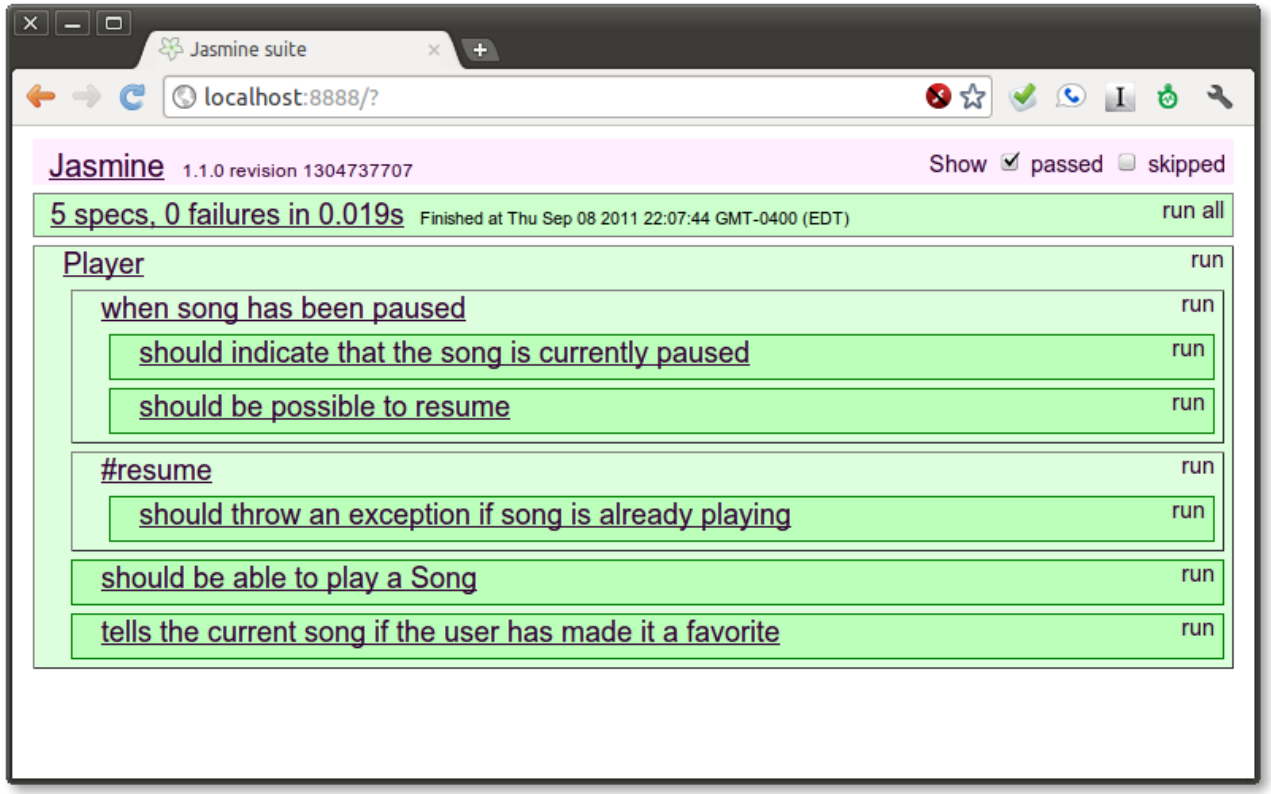
- JavaScript / client-side programming is lagging behind:
  - Tooling
  - Patterns
  - Wide-spread practices

GitHub and Travis-CI integration



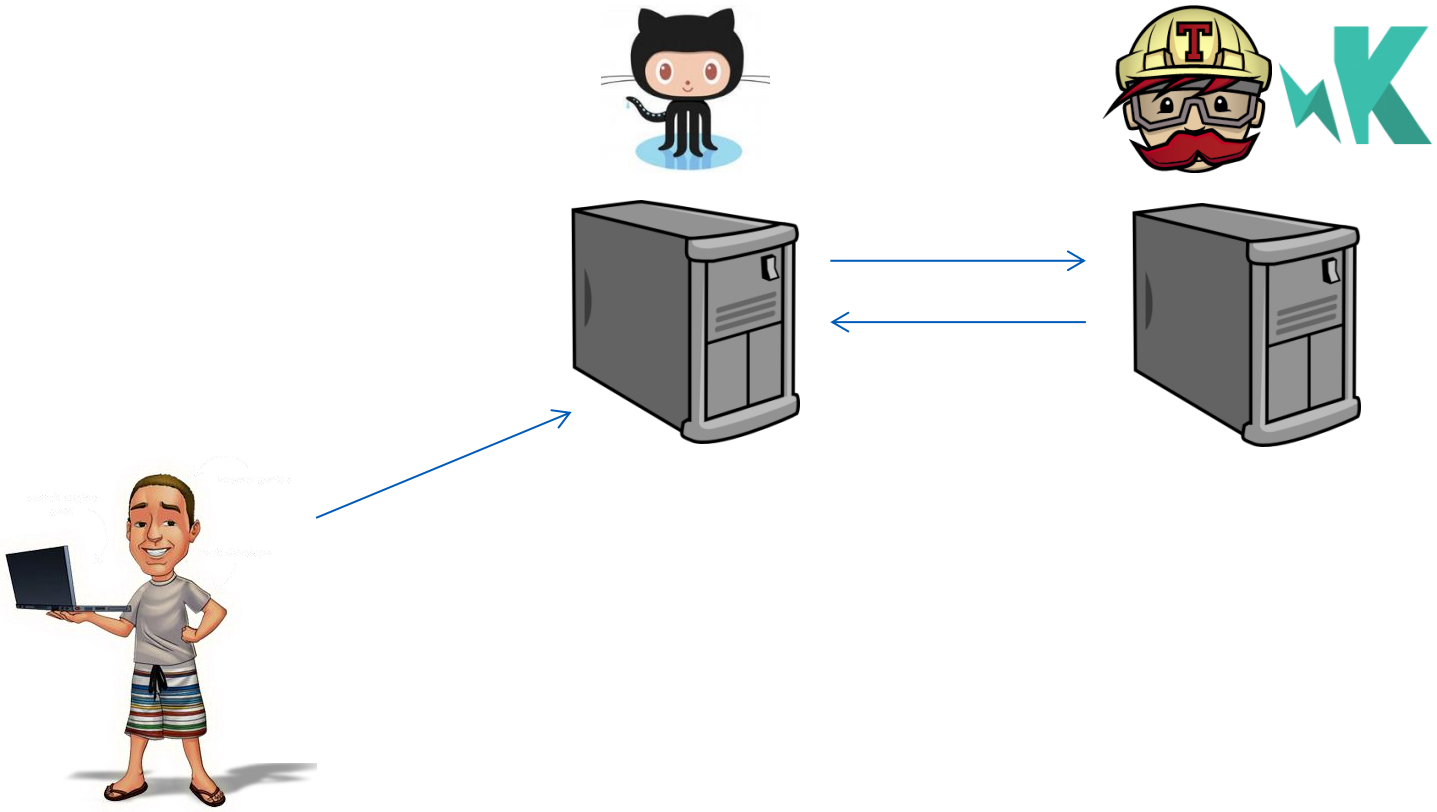
KARMA

Karma runner to the rescue



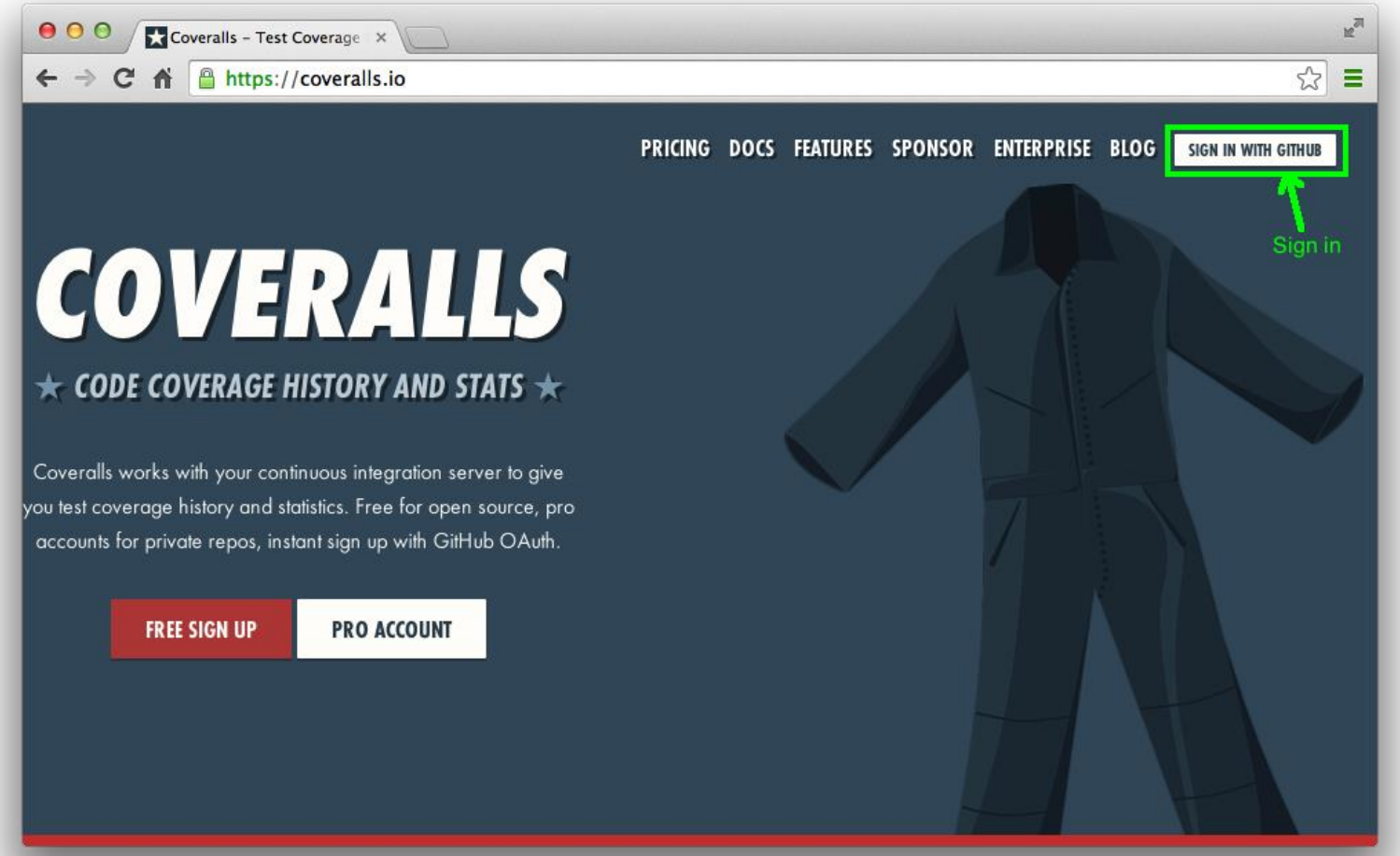
- Run unit tests in real browsers
- Execute the same tests in multiple browsers
- Allow easy integration with CI server, IDEs, command-line etc.

Travis-CI + Karma runner integration

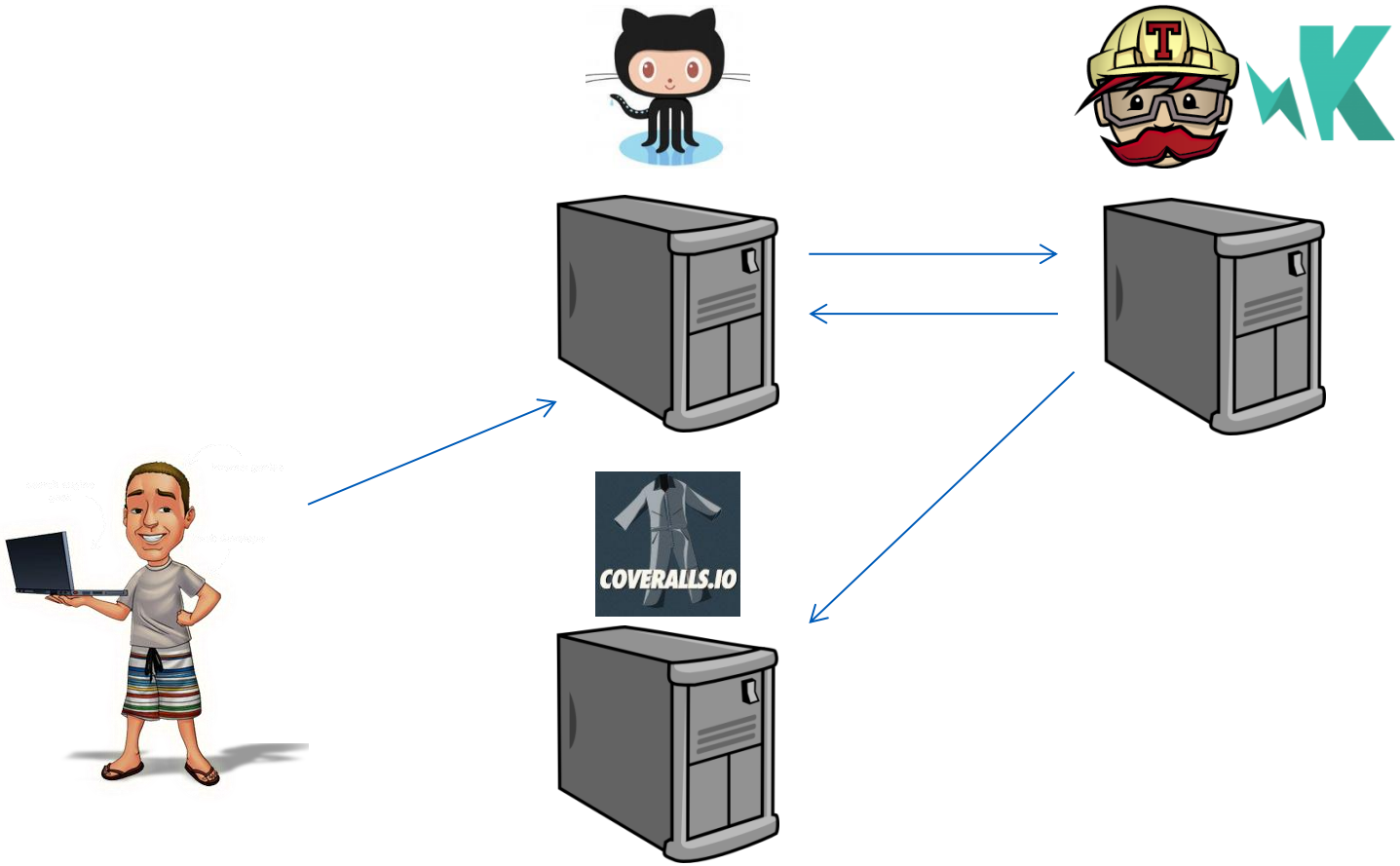


Reporting on code coverage with Coveralls

https://coveralls.io/

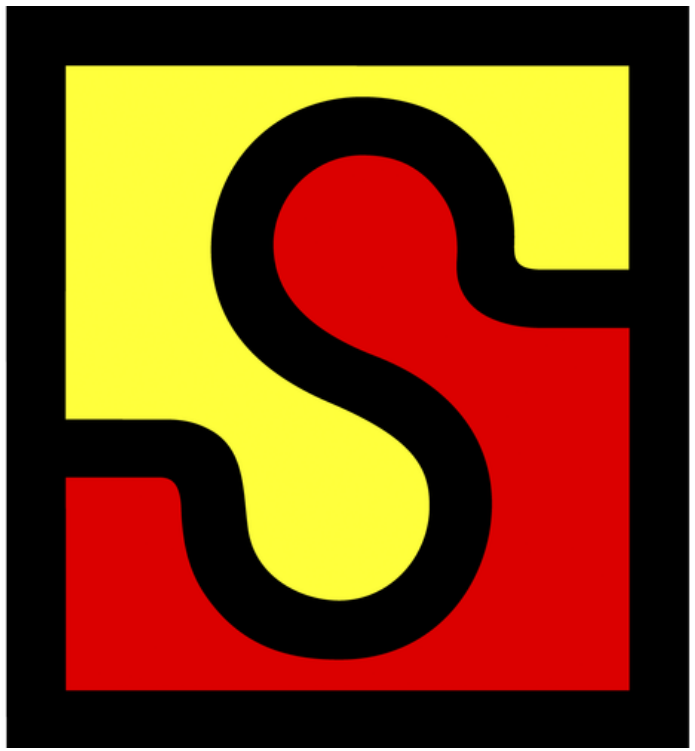


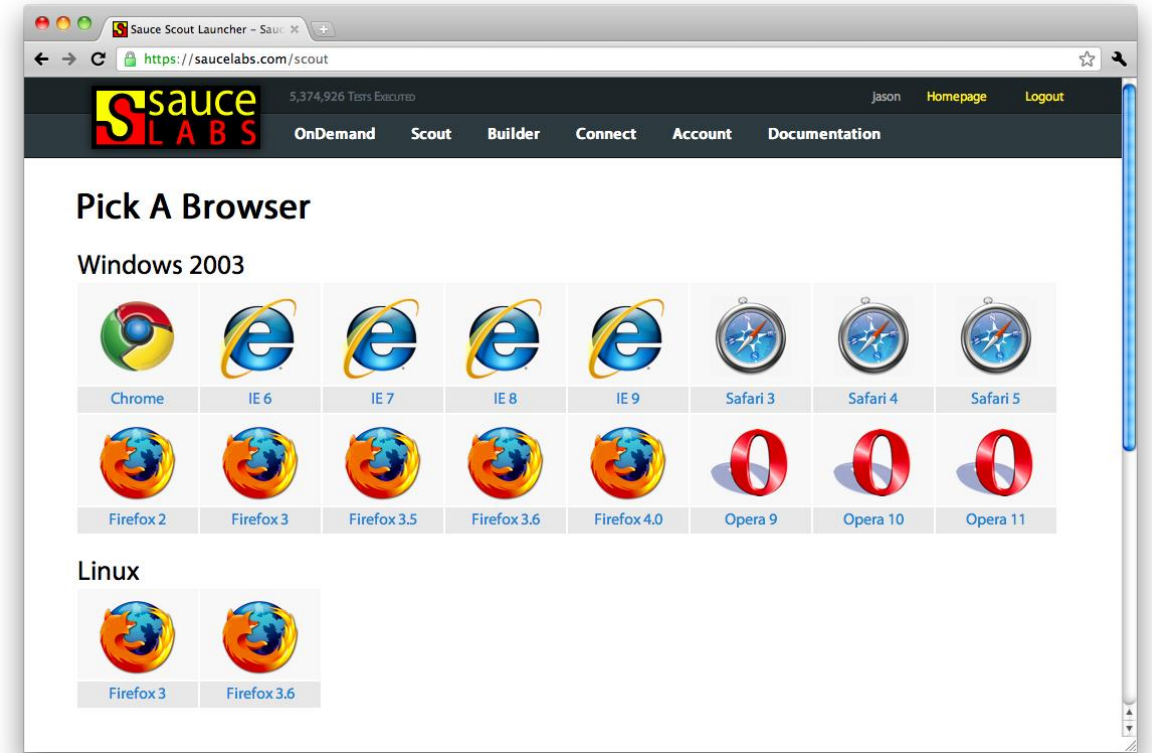
Travis-CI + Karma + Coveralls.io integration



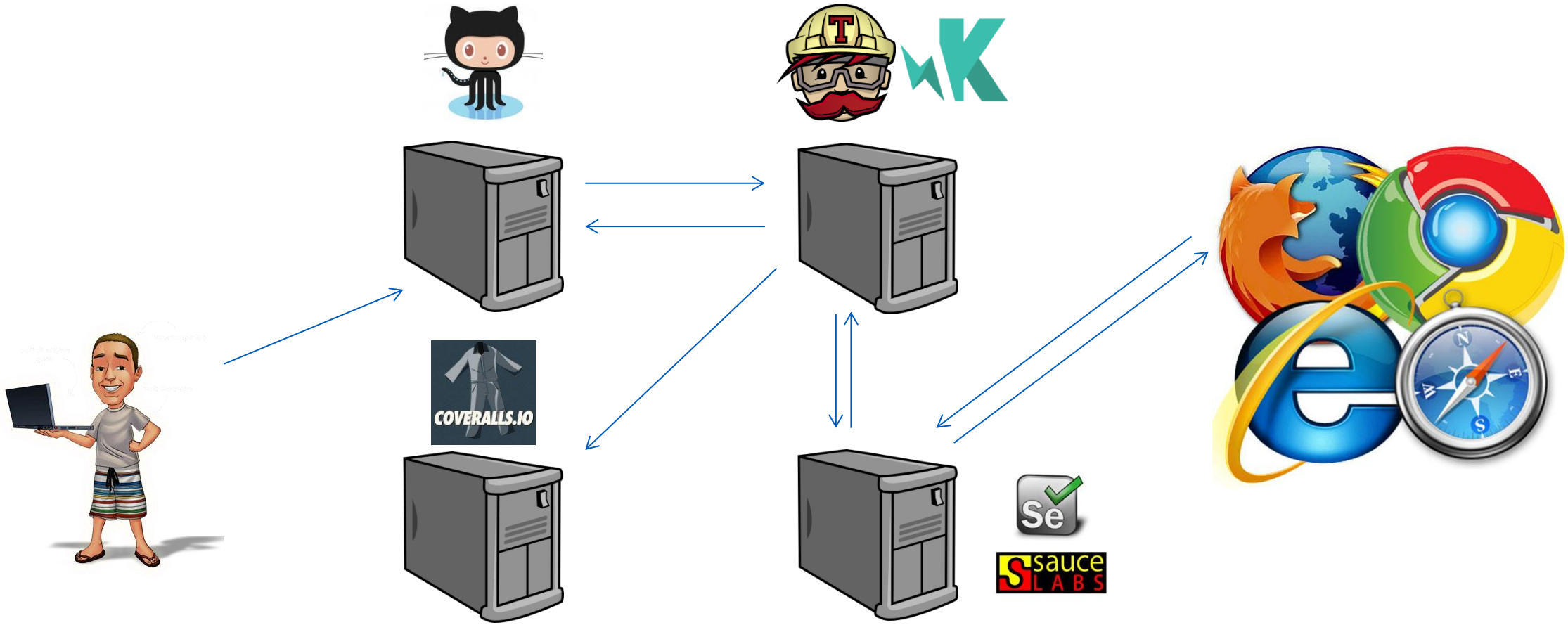
Multi-browser setup with SauceLabs

https://saucelabs.com/platforms





Travis-CI + Karma + SauceLabs integration



Future possibilities are endless

Example: https://david-dm.org/

"What gets measured
gets managed"

- Tracking dependencies status
- Deliverables size
- Commits per file
- Etc.

Takeaways

Takeaways

- We need to test our software before shipping it to the customers
- Humans are too creative to do work that machines can do – automate everything you can
- Client-side code is hard to test but the OpenSource community shows us tools and practices that we can use today

Bits and pieces

- https://github.com/
- https://travis-ci.org/
- http://karma-runner.github.io/
- https://coveralls.io/
- https://saucelabs.com/
- https://david-dm.org/

Thank you