Mutation Testing in Angular with Stryker

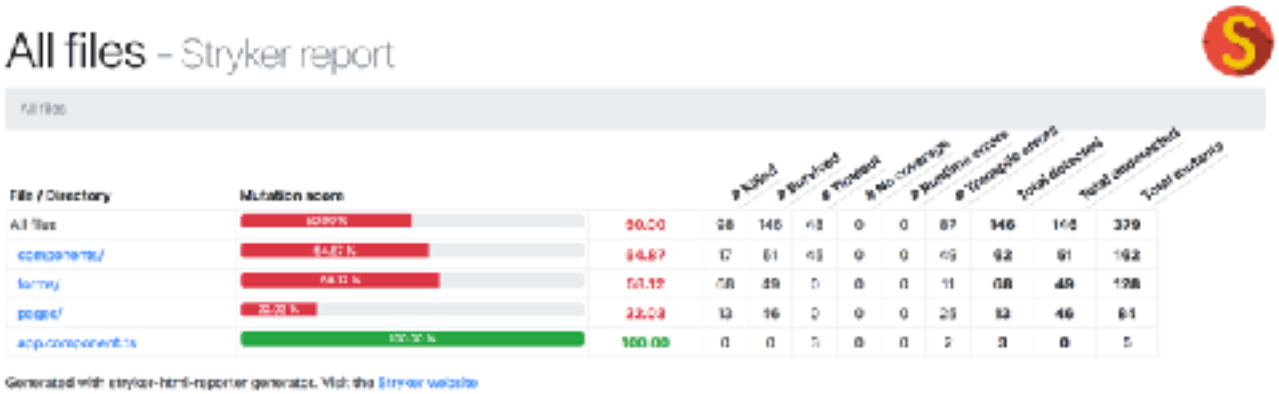
What is Mutation testing?
Mutation testing is the injection of deliberate faults in your code in order to ‘test your tests’. In short, if your tests are to succeed after the injection of ‘mutants’ then one of two things are happening: either your tests are not written properly, or your code is not necessary. In theory, well written code and tests should mean that any changes to the code will cause it to fail. An example of a mutation would be to change logic (eg. > to < or true to false). If your code continues to pass all tests, this mutant is said to have survived and therefore lowers the overall ‘score’ for your tests.

The concept of mutation testing is becoming. A more popular way to measure the quality of the tests within an application. In previous times one would use code coverage analysis to measure this quality, however this does little to ensure the quality of the testing; it is easy to simply fake this by touching every line of the code and ensuring tests all return true. This is an issue given many companies have non-functional requirements within their IT organisation to have their applications tested to a specific level of code coverage, occasionally affording the ability for developers to ‘cheat’ these requirements.

The concept of mutation testing has been around for some time, however until recently has been unpopular due to the large amount of re-compiling and re-running of tests required (the full test package must be run for each mutant). However with the advancement of computational power and introduction of technologies such as Red Hat’s OpenShift it is now easy to run mutation testing within a CI system.

Mutation Testing with Stryker

For this instance we have implemented the Stryker [https://stryker-mutator.github.io/]
mutation testing framework, one of the most mature Javascript mutation testing frameworks available. The framework supports a number of different mutation types such as changing math operators, logic operators, removing conditionals and removing entire block statements. A full list of mutants can be found here [https://stryker-mutator.github.io/mutators.html].



Setting up Stryker for an Angular app with Jenkins

In order to set up Stryker with an Angular app, it can be slightly more difficult than a more vanilla JS application. If Webpack is being used, further additions are needed. For this case I have used Webpack. In order to set up it is possible to follow Stryker’s easy to sue CLI, however it was not evident how to set up an

application using Angular and Webpack. As such it is ideal to use the config files given below in order to speed up setup. Our application is also running Jasmine and Karma, therefore any alterations from this will require a change to the config.

The config files for Styker can be found at *https://github.com/Tompage1994/stryker-config*

The next step is to install Stryker and other required dependencies. For our application these are:

```
"stryker": "^0.22.1",
"stryker-api": "^0.16.0",
"stryker-html-reporter": "^0.13.2",
"stryker-jasmine": "^0.8.2",
"stryker-karma-runner": "^0.14.3",
"stryker-typescript": "^0.10.1",
"stryker-webpack-transpiler": "^0.3.2”
```

In order to achieve this run:

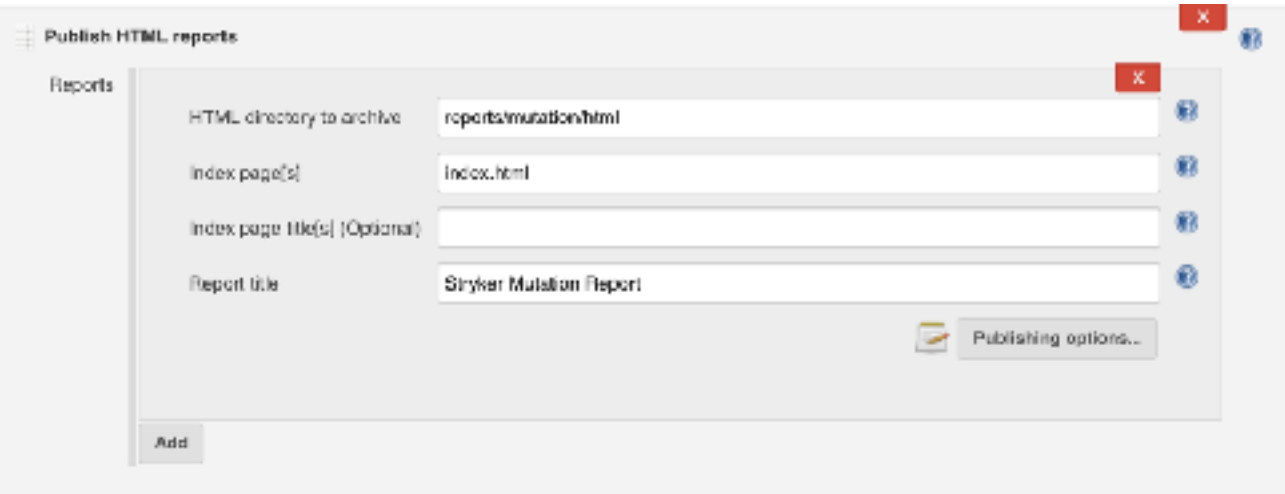
```
npm install --save-dev stryker styker-api stryker-html-reporter stryker-jasmine
stryker-karma-runner stryker-typescript stryker-webpack-transpiler
```

Running Stryker Locally

It is very simple to run the Stryker mutation test locally, however it may take a long time to run depending on the size of your code base. (Running on our medium sized application takes approximately 30 minutes). Once everything is set up properly it should be possible to run `` `stryker run` `` in order to begin the process. If trouble is occurring it may be ideal to add a flag `` `–logLevel debug` `` to this command to enable additional logs.

Integrating with Jenkins

The easiest way to allow Jenkins to run the Stryker tests is to add `"stryker": "stryker run”`, to the scripts section of your *package.json* in order to allow Jenkins to simply run ```` ```npm run stryker``` ````. The config file provided also allows html reporting from the run. This means it is possible to use the HTML Publisher plugin of Jenkins in order to present the results of the test.



In our version we have not done anything special but run Stryker on an NPM enabled Jenkins slave [[https://github.com/openshift/jenkins/blob/master/agent-nodejs-8/Dockerfile.rhel7]]. With our application we are able to run the slave in 40 minutes. This is clearly not ideal for a full end to end pipeline so we currently only run Stryker once per day (overnight) however with a little further work it may be possible to scale up the slave builds and run the tests in parallel, a major feature of Stryker which runs each mutation test on a different core of your system. The fact the Jenkins slave is a Kubernetes pod should mean it is easy to use OpenShift to orchestrate pod scaling to further the parallelism to a point we may be able to run a Stryker test as part of a full promotion pipeline, preventing upgrading to production if the code quality is too low.

Next Steps

After analysing mutation testing through Stryker, I have come to the conclusion that code coverage; which is used by so many companies, is becoming obsolete due to the inability to properly measure the quality of the codebase. Code mutations have been dubbed a way to 'test your tests' and from experience this seems to be the case.

Although there are still some flaws with the relatively new Stryker framework, advancements are coming. Only recently typescript support was added, allowing testing of Angular applications. Further developments seem to be coming and the team are open to issues being submitted at their GitHub repository *https://github.com/stryker-mutator/stryker* or even code contributed.