Testing Policy - Docks

TripleParity

Contents

1 Testing Process

1.1 Peer Review

Before code can be merged into the develop branch a Pull Request has to be created. Three reviews are required from other developers before the pull request can be merged into develop.

During peer reviews the following should be checked:

- Architectural problems - will this cause problems in the future?

- Compliance with requirements and design - is that what we need?

- Coding Standards - proper code formatting and security standards?

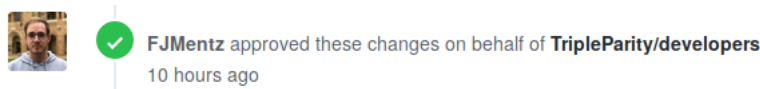Figure 1: At least 3 peer reviews are needed for a feature to be merged

Figure 2: An example of a positive peer review

1.2 Automated Testing and Continuous Integration

When a commit is made to the 'docks-ui' repository on GitHub two processes are started:

- Travis CI Builds the repository

- Docker Hub builds the repository and creates a Docker Image that can be deployed in production

1.2.1 Travis CI

The history of test reports for Travis CI can be viewed at https://travis-ci.org/TripleParity/docks-ui/branches. If the build was not successful it will be marked as 'failed'

Travis CI is used for running tests associated with each repository.

For the frontend Angular generates a set of tests for each component to verify that the component was successfully created. These tests run inside a headless (no screen required) Chrome browser.
 The backend also has a suite of unit tests that are executed on Travis CI.

Default Branch

| ✓ master | # 48 passed | ⟜ ff703e7 ⧉ | ✓ | | | | |
| 1 build | 27 days ago | 🔴 Evert Geldenhuys | | | | | |

Active Branches

∞ networks2	# 263 started	⟜ 10affa1 ⧉	⚬	✓	✗		
3 builds	-	Ⓒ CDuPlooy					
✗ loginPage	# 262 failed	⟜ 95d81d1 ⧉	✗	✗			
2 builds	19 minutes ago	Ⓒ Paul Wood					
✓ develop	# 261 passed	⟜ 8119ea9 ⧉	✓	✓	✓	✓	✓
19 builds	29 minutes ago	Ⓒ GitHub					

Figure 3: Screenshot of Travis CI branch build history

Figure 4: 18 UI tests running on Travis

1.2.2 Docker Cloud

Docker images are build on Docker Cloud. These image can then be deployed in a production environment or for development and testing. Images can be viewed at https://hub.docker.com/u/tripleparity/
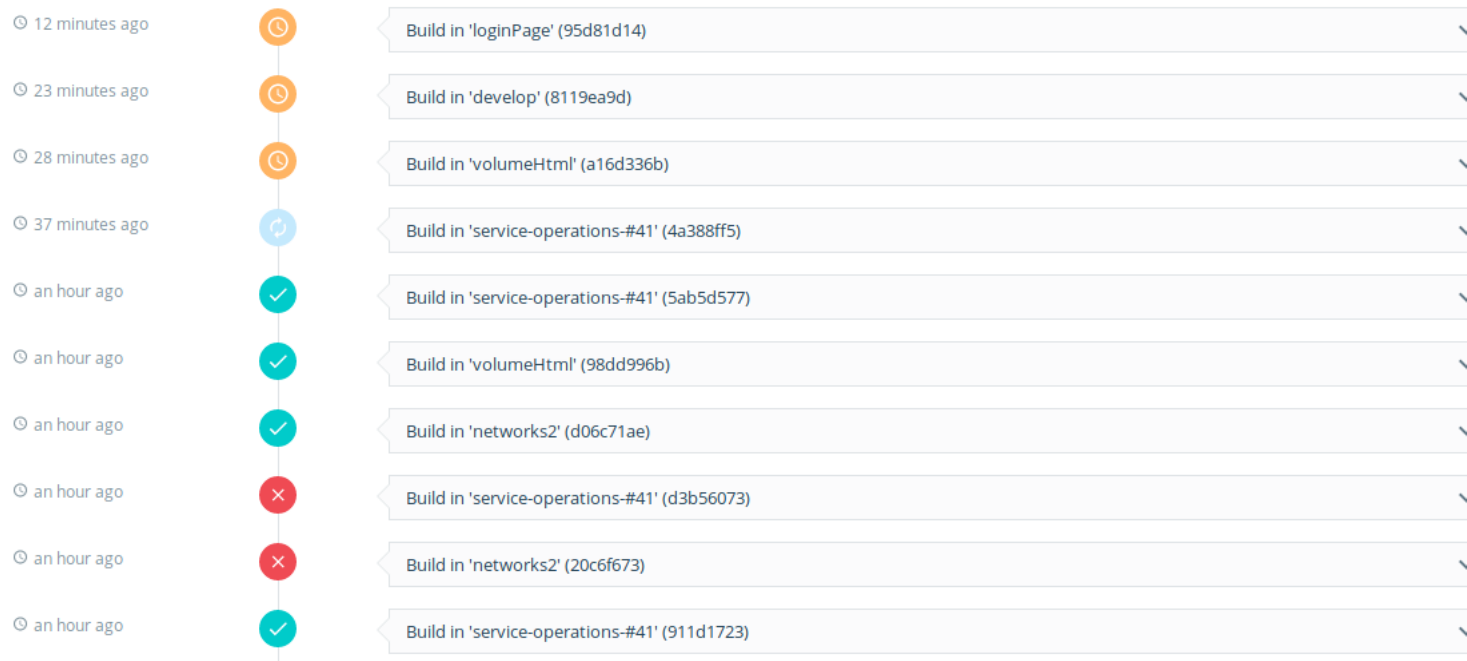
Figure 5: Screenshot of Docker Cloud Build History

1.3 Docks UI

1.3.1 tslint

Inside of the UI we aditionally use tslint, protractor and karma for testing.
Tslint is a static analysis tool to check typescript code for readability, maintainability and functionality errors.
It can be customised with custom linting rules and formatters. This can help catch type errors in the code.


```
fetchVolumes() {
  this.volumeService.getVolumes().subscribe(
    (volumes: Volume[]) => {
      this.volumes = [1, 2, 3];
      this.searchString = [...volumes];
      this.isLoaded = true;
    },
    (err: VolumeError) => {
      this.toastr.error(err.message, 'An error occured');
    }
  );
}
```

Figure 6: Screenshot of tslint linting

In the above example an array of numbers is mistakenly assigned to an array of Volumes. The linter clearly highlights this as a problem.

4

1.3.2 karma

Karma is a tool which creates easy testing environments for developers; It's main goal is to give developers instant feedback about their test cases.

```
module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular/cli'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage-istanbul-reporter'),
      require('@angular/cli/plugins/karma')
    ],
    client:{
      clearContext: false // leave Jasmine Spec Runner output visible in browser
    },
    coverageIstanbulReporter: {
      reports: [ 'html', 'lcovonly' ],
      fixWebpackSourcePaths: true
    },
    angularCli: {
      environment: 'dev'
    },
    reporters: ['progress', 'kjhtml'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['ChromeHeadless'],
    customLaunchers: {
      ChromeHeadlessNoSandbox: {
        base: 'ChromeHeadless',
        flags: ['--headless', '--disable-gpu','--disable-translate', '--disable-extensions', '--remote-debugging-port=9223',
      }
    },
    singleRun: true
  });
};
```

Figure 7: Screenshot of karma config

In particular we run a headless instance of chrome with all extensions disabled to test our code; Istanbul is also used to generate a code coverage report. The test cases are run only once but can be configured such that the tests are run every time a file is changed.
When a commit is made to a branch, these tests are also run.

1.3.3 Protractor

Protractor is an end-to-end test framework. Tests are run in a browser which automatically interacts with the page as a user would

1.4 Docks API

1.4.1 eslint

eslint is essentially the same as tslint but for javascript.

1.4.2 jest

jest is a javascript testing framework. It has some nice features such as instant feedback meaning failed tests run first. This means we can easily tend to issues.

Jest is used in the backend to run tests on the api itself.

```javascript
describe('Authentication', function() {
  it('should not serve Docker API requests without a JWT', function() {
    return frisby.get(host + '/docker/containers/json').expect('status', 401);
  });

  it('should serve JWT for valid ("admin/admin") credentials', function() {
    return frisby
      .post(tokenURL, credentials)
      .expect('status', 200)
      .expect('bodyContains', 'jwt')
      .expect('jsonTypes', 'jwt', Joi.string().required())
      .then(function(res) {
        let jwt = res.json['jwt'];

        return frisby
          .fetch(host + '/', {
            method: 'GET',
            headers: {
              Authorization: 'Bearer ' + jwt,
            },
          })
          .expect('status', 200)
          .expect('bodyContains', 'Welcome to Express');
      });
  });
});
```

Figure 8: Screenshot of jest test

1.4.3 codeclimate

An automated code review tool.

Tests can be seen here.

```javascript
describe('Authentication', function() {
  it('should not serve Docker API requests without a JWT', function() {
    return frisby.get(host + '/docker/containers/json').expect('status', 401);
  });

  it('should serve JWT for valid ("admin/admin") credentials', function() {
    return frisby
      .post(tokenURL, credentials)
      .expect('status', 200)
      .expect('bodyContains', 'jwt')
      .expect('jsonTypes', 'jwt', Joi.string().required())
      .then(function(res) {
        let jwt = res.json['jwt'];

        return frisby
          .fetch(host + '/', {
            method: 'GET',
            headers: {
              Authorization: 'Bearer ' + jwt,
            },
          })
          .expect('status', 200)
          .expect('bodyContains', 'Welcome to Express');
      });
  });
});
```

Figure 9: Screenshot of codeclimate score