

---

## grunt-perfbudget

Grunt task for Performance Budgeting

### Performance budgeting thanks to the magic of WebPageTest

grunt-perfbudget is a Grunt.js task for enforcing a performance budget (more on performance budgets). It uses the wonderful [webpagetest.org](http://webpagetest.org) and the WebPageTest API Wrapper for NodeJS created by Marcel Duran.

grunt-perfbudget uses either a public or private instance of WebPageTest to perform tests on a specified URL. It compares test results to budgets you specify. If the budget is met, the tasks successfully completes. If it the page exceeds your performance budgets, the task fails and informs you why.

### Getting Started

This plugin requires Grunt ~0.4.1

If you haven't used Grunt before, be sure to check out the Getting Started guide, as it explains how to create a Gruntfile as well as install and use Grunt plugins. Once you're familiar with that process, you may install this plugin with this command:

```
1 npm install grunt-perfbudget --save-dev
```

Once the plugin has been installed, it may be enabled inside your Gruntfile with this line of JavaScript:

```
1 grunt.loadNpmTasks('grunt-perfbudget');
```

### The “perfbudget” task

#### Required configuration properties

While grunt-perfbudget provides defaults for most configurable options, it does require the URL to be tested, as well as an API key to use if testing against the public instance of WebPageTest. For more information on obtaining a key, see this thread on the WebPageTest forums.

These can be set in your Gruntfile.js config file like so:

```
1 perfbudget: {  
2   default: {
```

---

```
3     options: {
4       url: 'http://google.com',
5       key: 'API_KEY_HERE'
6     }
7   }
8 }
```

With these configuration properties set, you can add `perfbudget` to your default tasks list. That'll look something like this:

```
1 grunt.registerTask('default', ['jshint', 'perfbudget']);
```

With this in place, grunt-perfbudget will now test your site against the default performance budget settings to see if you're passing.

## Options

grunt-perfbudget takes the following options:

**options.output** Type `String` Default value: NONE

The file to output the JSON results to.

**options.url** Type `String` Default value: NONE

The url you want to perform the tests on.

**options.key** Type `String` Default value: NONE

The API Key for the public instance of WPT. *Not needed if using a private instance of webpagetest*

**options.location** Type `String` Default value: 'Dulles\_Nexus5'

The default WPT location/device to conduct the test using.

**options.wptInstance** Type `String` Default value: 'www.webpagetest.org'

The WPT instance to conduct the tests with.

**options.pollResults** Type `Number` Default value: 5

The frequency (in seconds) to poll for results after the test has been scheduled.

---

**options.timeout** Type `Number` Default value: 60

Timeout (in seconds) for the tests to run.

**options.connectivity** Type `String` Default value: NONE

The connectivity profile to use. WPT provides the following options: Cable, DSL, FIOS, Dial, 3G, Native, custom.

**options.bandwidthDown** Type `String` Default value: NONE

The download bandwidth in Kbps. *Used when connectivity is set to `custom`.*

**options.bandwidthUp** Type `String` Default value: NONE

The upload bandwidth in Kbps. *Used when connectivity is set to `custom`.*

**options.latency** Type `String` Default value: NONE

The RTT latency in milliseconds. *Used when connectivity is set to `custom`.*

**options.packetLossRate** Type `String` Default value: NONE

The package loss rate (percentage of packets to drop). *Used when connectivity is set to `custom`.*

**options.repeatView** Type `Boolean` Default value: `false`

If set to `true`, tests the budget against the repeat view. *By default, `perfbudget` tests the budget against the first view and doesn't ask WPT to run a test on the repeat view.*

**options.login** Type `String` Default value: NONE

Username for authenticating tests.

**options.password** Type `String` Default value: NONE

Password for authenticating tests.

**options.authenticationType** Type `Number` Default value: 0

Type of authentication. 0 = Basic, 1 = SNS.

---

**options.runs** Type `Number` Default value: 1

Number of test runs. *If the test is run more more than once, the budget is tested against the median result of the runs.*

**options.budget** Type `Object`

Allows you to specify a performance budget. *If the test is run more more than once, the budget is tested against the median result of the runs.*

The variables you can use as a budget include:

**budget.visualComplete** Type `String` Default value: NONE

The budget for visually complete in milliseconds.

**budget.render** Type `String` Default value: 1000

The budget for start render time in milliseconds.

**budget.loadTime** Type `String` Default value: NONE

The budget for load time in milliseconds.

**budget.docTime** Type `String` Default value: NONE

The budget for `document.complete` in milliseconds.

**budget.fullyLoaded** Type `String` Default value: NONE

The budget for fully loaded time in milliseconds.

**budget.bytesIn** Type `String` Default value: NONE

The budget for overall weight in bytes.

**budget.bytesInDoc** Type `String` Default value: NONE

The budget for number of bytes downloaded before the Document Complete time.

---

**budget.requests** Type `String` Default value: NONE

The budget for overall number of requests.

**budget.requestsDoc** Type `String` Default value: NONE

The budget for number of requests made before the Document Complete time.

**budget.SpeedIndex** Type `String` Default value: 1000

The budget for calculated SpeedIndex.

**budget.userTime** Type `String` Default value: NONE

The budget for the final user timing mark recorded on the page.

You can test against a specific user timing mark like so:

```
1 perfbudget: {
2   default: {
3     options: {
4       url: 'http://google.com',
5       key: 'API_KEY_HERE',
6       budget: {
7         'userTime.CUSTOM_MARK': '1500'
8       }
9     }
10  }
11 }
```

For more information on User Timing, see <http://www.w3.org/TR/user-timing/> ### Usage Examples

#### 1. Test <http://google.com> against default budget settings

```
1 perfbudget: {
2   default: {
3     options: {
4       url: 'http://google.com',
5       key: 'API_KEY_HERE'
6     }
7   }
8 }
```

#### 2. Test <http://google.com> using custom budget for SpeedIndex and Visually Complete

```
1 perfbudget: {
2   default: {
3     options: {
4       url: 'http://google.com',
```

---

```
5     key: 'API_KEY_HERE',
6     budget: {
7         visualComplete: '4000',
8         SpeedIndex: '1500'
9     }
10 }
11 }
12 }
```

### 3. Test URL using custom budget and private WPT Instance

```
1 perfbudget: {
2     default: {
3         options: {
4             url: 'http://google.com',
5             wptInstance: 'http://PRIVATE_INSTANCE.com',
6             budget: {
7                 visualComplete: '4000',
8                 SpeedIndex: '1500'
9             }
10         }
11     }
12 }
```

### 4. Test http://google.com using a custom budget against the median result of 5 test runs.

```
1 perfbudget: {
2     default: {
3         options: {
4             url: 'http://google.com',
5             key: 'API_KEY_HERE',
6             runs: 5,
7             budget: {
8                 visualComplete: '4000',
9                 SpeedIndex: '1500'
10            }
11        }
12    }
13 }
```

### 5. Test http://google.com against default budget settings and output the results to a file.

```
1 perfbudget: {
2     default: {
3         options: {
4             url: 'http://google.com',
5             key: 'API_KEY_HERE',
6             output: 'wpt-results.json'
7         }
8     }
9 }
```

---

## Contributing

In lieu of a formal styleguide, take care to maintain the existing coding style. Add unit tests for any new or changed functionality. Lint and test your code using Grunt.

## Release History

- Version 0.1.3: Bug fix for custom options. Now includes ability to use HTTP authentication on tests.
- Version 0.1.4: Ability to define custom number of test runs.
- Version 0.1.5: Improved polling using the underlying API. Users can now set polling frequency as well as a timeout for tests.
- Version 0.1.6: Ability to test budget against repeat views; minor bug fixes; better error handling.
- Version 0.1.7: Improved error handling.
- Version 0.1.8: Ability to output results to JSON; bug fixes; improved error handling.
- Version 0.1.9: Security improvements. Also changing default instance of WPT to use https now that the site supports it.