Continuous integration hacks for Angular with Jenkins

@AlainChautard - angulartraining.com

About me - Alain Chautard (or just Al)

Google Developer Expert in Web technologies / Angular

Java developer since 2006

Angular JS addict since 2011

Organizer of the Sacramento Angular Meetup group

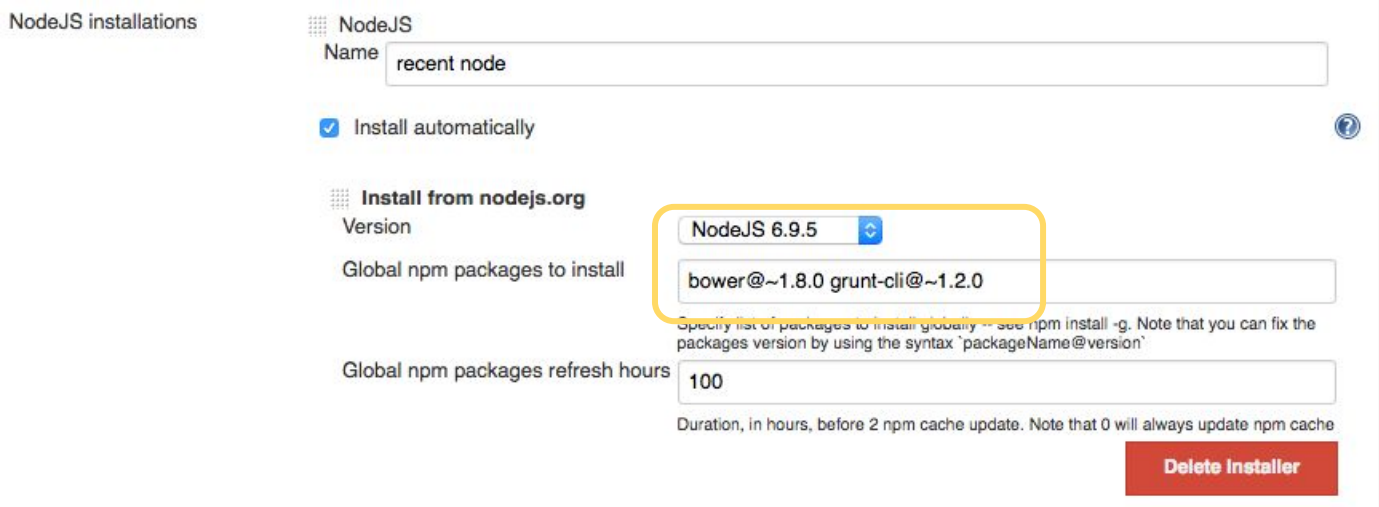
Web consultant (60%) / trainer (40% of the time) @ angulartraining.com

To do list for CI with Angular and Jenkins

- Have Node.JS and Angular CLI installed
- Run **ng test** and **ng e2e** to make sure all is good
- Run **ng build --prod** or anything close to it (AOT build)
- Make sure your app size is under control
- On success, deploy your build artifacts

Have Node.JS and Angular CLI installed

- Use the NodeJS plugin for Jenkins: Multiple version support as well as global npm packages like Angular CLI
- This plugin can be found at: https://wiki.jenkins.io/display/JENKINS/NodeJS+Plugin



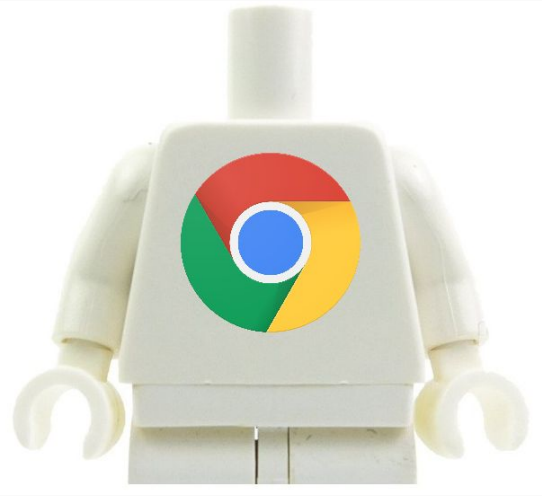
Run `ng test` and `ng e2e` to make sure all is good

- The NodeJS plugin gets Angular CLI installed for us: Running these two commands is not an issue
- The main problem then is… Those tests need a browser to run!
- And most CI servers use Linux, which means:
  - No Google Chrome available
  - No UI to launch a browser
- So what are our options?

Run `ng test` and `ng e2e` to make sure all is good

- **Solution:** Tests can be run with a UI-less browser
- Two main options:

Chrome Headless

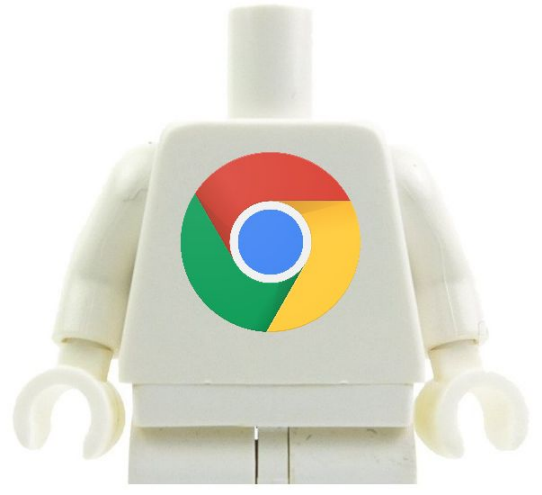


Phantom JS

Run `ng test` and `ng e2e` to make sure all is good

- **Chrome Headless** relies on a simple flag in Karma config:

```
customLaunchers: {
 ChromeHeadless: {
   base: 'Chrome',
   flags: [
      '--headless','--disable-gpu',
      '--no-sandbox',
      '--remote-debugging-port=9222']
 }
},
browsers: ['ChromeHeadless'],
singleRun: true
```



Run `ng test` and `ng e2e` to make sure all is good

- **Phantom JS** requires a specific Karma launcher (to be installed with npm)

```
plugins: [
 require('karma-jasmine'),
 require('karma-chrome-launcher'),
 require('karma-phantomjs-launcher'),
 require('karma-jasmine-html-reporter'),
 ...
],
…,
browsers: ['PhantomJS'],
singleRun: true
```

Generate a code coverage report

- Instead of running just **`ng test`**, use:

```
ng test --code-coverage
```

- This will generate a coverage report for your project:

All files

89.47% Statements 34/38 **66.67%** Branches 2/3 **55.56%** Functions 5/9 **87.5%** Lines 28/32

File		Statements		Branches		Functions		Lines	
src		100%	16/16	100%	0/0	100%	1/1	100%	16/16
src/app		84.62%	11/13	100%	0/0	60%	3/5	77.78%	7/9
src/app/book-list		77.78%	7/9	66.67%	2/3	33.33%	1/3	71.43%	5/7

Run `ng build --prod` or anything close to it (AOT build)

- Why run an AOT build?
  - To validate your HTML templates
  - To make sure your app is performant
  - To get an idea of how big your application is

Make sure your app size is under control

- Use budgets to make the build fail if too many dependencies get added:

```
"configurations": {
 "production": {
      "budgets": [
      {
        "type": "bundle",
        "name": "vendor",
        "baseline": "750kb",
        "warning": "100kb",
        "error": "200kb"
      }
    ],
```

Build fails is vendor bundle size > 950kb

Warning if > 850 kb and < 950 kb

How to build for different environments?

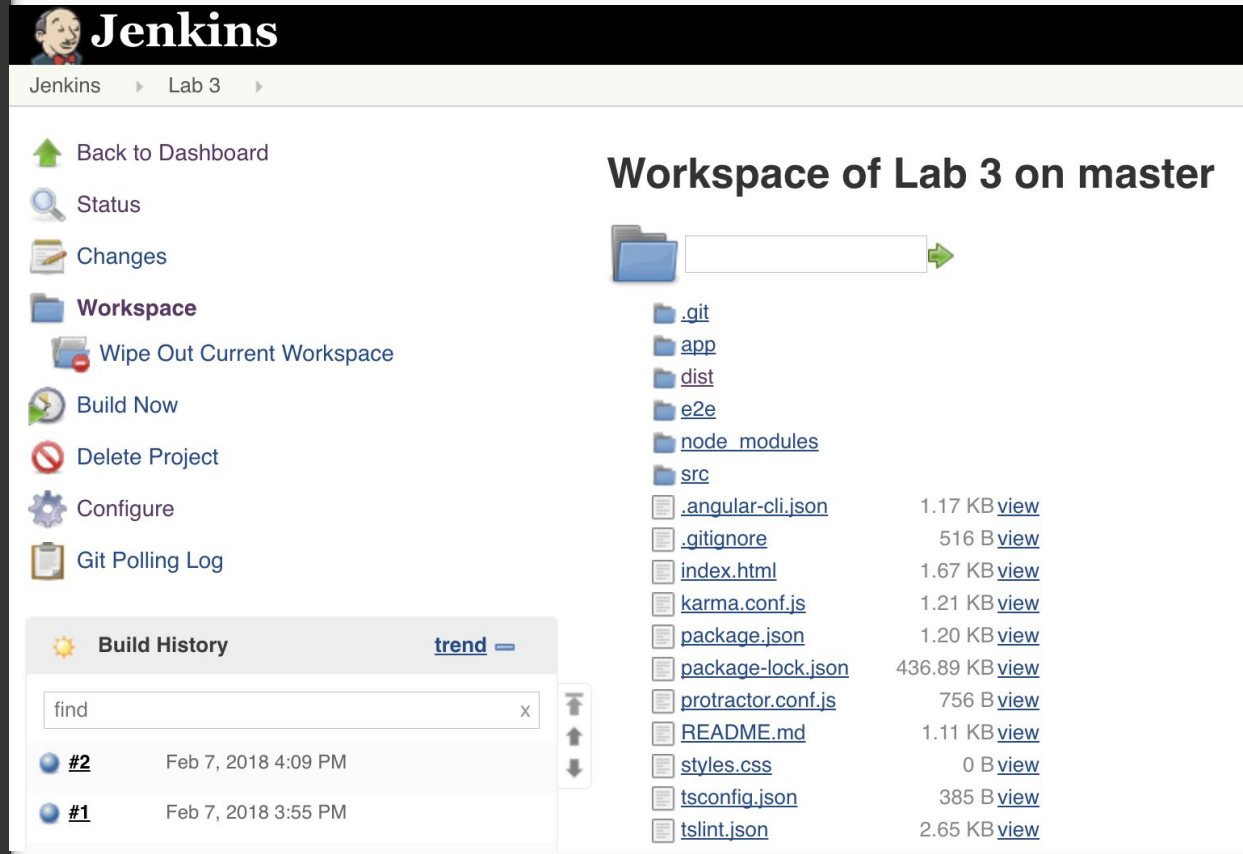
- You might need different builds (QA, prod, test, dev)
- Configurations are the solution for these distinct builds:

```
"configurations": {
 "production": {
        "fileReplacements": [
            { "replace": "src/environments/environment.ts",
              "with": "src/environments/environment.prod.ts"}
        ]
}, "qa": {
        "fileReplacements": [ …
              "with": "src/environments/environment.qa.ts"}
        ]
```

Build Artifacts

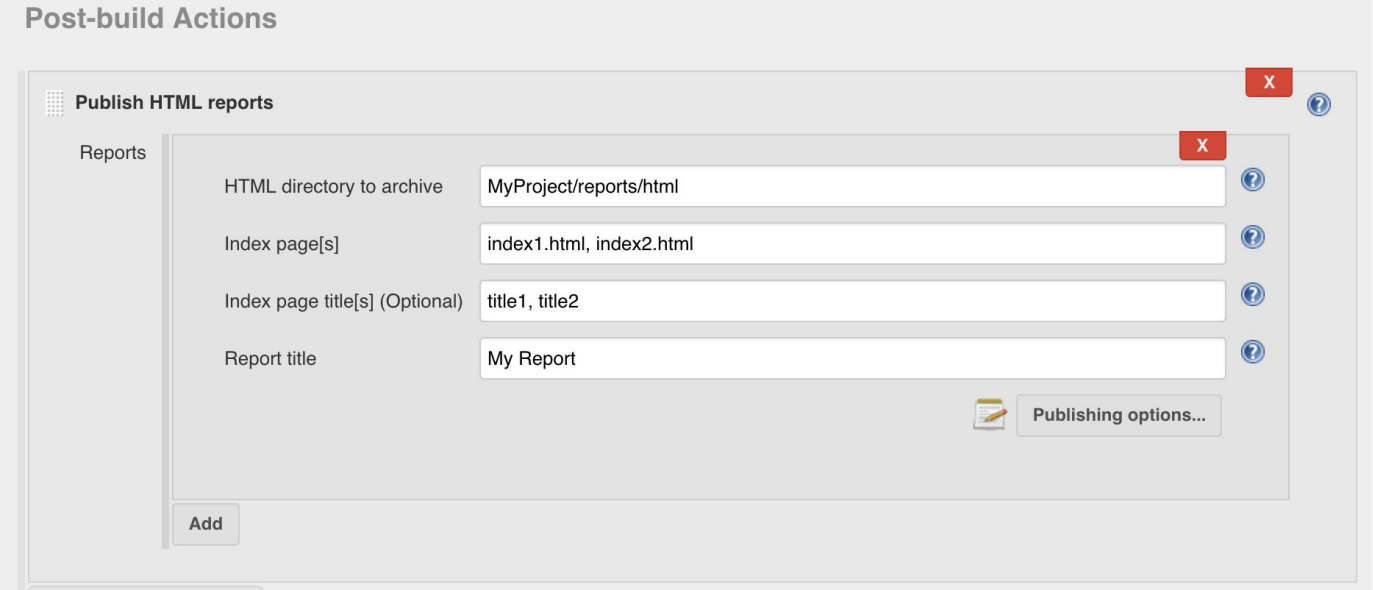
Build artifacts can be found in the workspace directory in Jenkins.

That's where builds can generate reports such as code coverage from **ng `test --code-coverage`**.



Make sure you test reports are easy to find

- HTML Publisher plugin creates links to reports created by your build.
- This plugin can be found at: https://wiki.jenkins.io/display/JENKINS/HTML+Publisher+Plugin



On success, deploy your build artifacts

- If you need to deploy your artifacts, a simple FTP / SCP / CP of the **`dist`** folder is all you need. Many plugins exist to do so:

Publish Over

Created by Bap bap2000, last modified on Mar 27, 2013

Goal

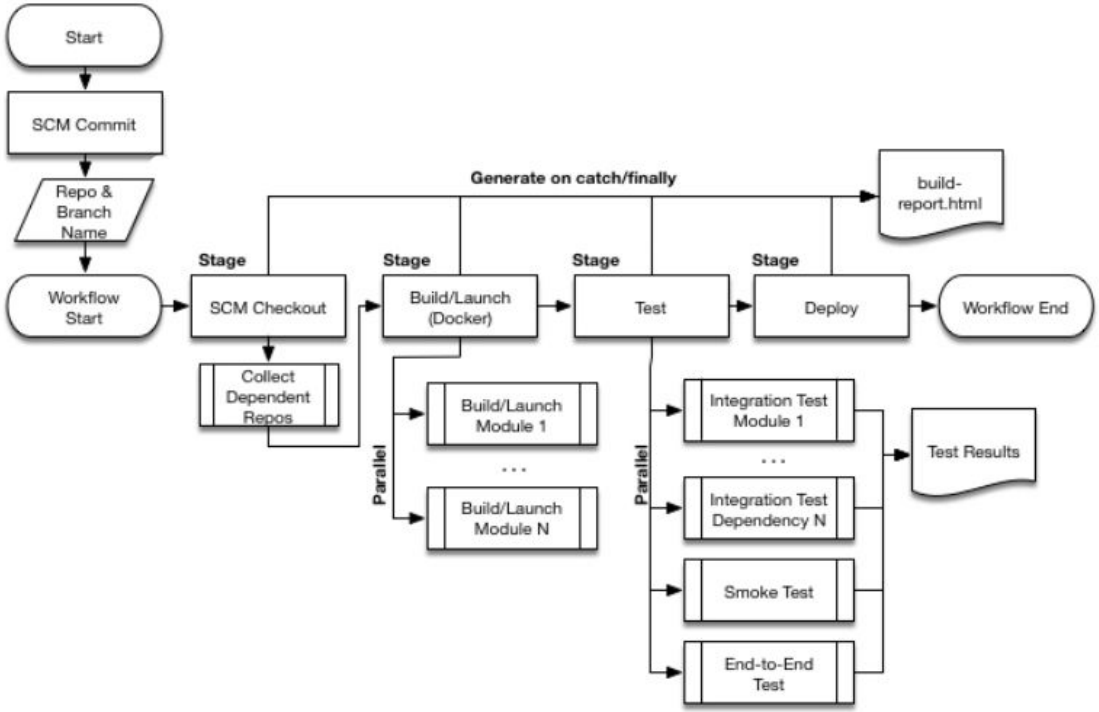
The goal of the Publish Over plugins is to provide a consistent set of features and behaviours when sending build artifacts ... somewhere.

Publish Over plugins

- Publish Over CIFS Plugin - send artifacts to a windows share
- Publish Over FTP Plugin - send artifacts to an FTP server
- Publish Over SSH Plugin - send artifacts to an SSH server (using SFTP) and/or execute commands over SSH

We now have a full continuous delivery process in place

- Congratulations! You now have a process to move your code from development to production



Thanks for your attention

Link to slides:

https://goo.gl/pmPZ91

My blog:

https://blog.angulartraining.com

Twitter:

@AlainChautard

ANGULAR TRAINING