Continuous Deployment from Github to WP Engine

Zach Watkins, WordPress Developer
AgriLife Communications

How many steps does it take to deploy one line change in your code?

Before:

1. Make a change locally
2. Run staging build tasks
3. FTP in to staging servers
4. Find where to upload the file
5. Upload changes for testing
6. If ready, run steps 1-5 for production
10. Push changes to Github repo
11. Ask users who clone our repo to run build scripts before using the plugin or theme

After

1. Make a change locally
2. Push to staging branch of repo
3. Test change on staging server
4. If ready, push to master branch
5. Double-check change on production server
6. Allow users to download the latest, ready-to-go release

Everything else is automatic!

Benefits

- Easier to upload bug fixes and improvements - everywhere
- Easier to maintain testing and production builds of our code
- Easier to release an end-user package of our themes and plugins
- Nearly eliminates the need to upload files through FTP

Case Study: AgriFlex3 Theme

1. JS and CSS built with Sass, Compass, Foundation, and Coffeescript
2. Separate staging and production tasks
3. Github releases built with grunt-contrib-compress, grunt-gh-release, and custom tasks
4. Present on 6 of our WP Engine installs
5. Used outside of our department

Compiling CSS

1. Sass files divided by purpose and theme variation (navigation, content, etc)
2. Foundation files and mixins imported as needed
3. Compass builds CSS files configured to the destination environment
    a. One file for each theme variation
    b. Staging receives expanded files with sourcemaps, which show you the file and line of a css rule’s source file when using inspector tools built into modern browsers
    c. Production receives compressed files
4. Sass linting is performed in the staging task
    a. Linting helps enforce code formatting standards by scanning code and alerting the user of violations

Compiling JS

More readable, manageable language that compiles to JavaScript

Nonessential currently, used to initialize Foundation

Releasing to Github with Grunt

- grunt-contrib-compress targets files for compression into AgriFlex3.zip
- Define an environmental variable using a Github personal access token
  - The token is user-specific, meaning all releases are published under that user's name
- Custom Grunt tasks pull the repository's commit history so proper attribution is made in the release message
- grunt-gh-release uses package.json, the access token variable, and the zip file to upload a release to the repository

Releasing to Github with Grunt

1.1.8

a37fc58

Release

ZachWatkins released this on Oct 13, 2016 · **40 commits** to master since this release

Zachary Watkins (2):

- Restore list style appearance to menu widgets
- Add menu widget styles to other variations

Downloads

AgriFlex3.zip	1.06 MB
Source code (zip)	
Source code (tar.gz)	

Releasing to Github with Grunt

grunt-contrib-compress targets files for compression into AgriFlex3.zip:

- Define the compress task (Gruntfile.coffee, line 64)
- File paths may use Grunt's globbing pattern rules provided by node-glob
  - {src: ['css/*.css']}
  - {src: ['img/**']}
  - {src: ['bower_components/foundation/{css,js}/**']}
  - {src: ['vendor/**', '!vendor/composer/autoload_static.php']}

Releasing to Github with Grunt

Define an environmental variable using a Github personal access token:

1. Go to https://github.com/settings/tokens and generate a new token scoped to public repositories (and private if necessary). Save this token temporarily.
2. Open your terminal and define the token as an environmental variable named RELEASE_KEY
    a. On Mac:
        i. $ cd ~ && nano .bash_profile
        ii. Add `export RELEASE_KEY=”yourkey”`, where yourkey is your new token, and exit
        iii. $ source ~/.bash_profile
    b. On Windows, go to Control Panel -> System -> Environment and define it with the GUI

Releasing to Github with Grunt

grunt-gh-release uploads a release to the repository

- Define the gh_release task (Gruntfile.coffee, line 89)
- Access the release key with process.env.RELEASE_KEY
- Use property values within strings by wrapping them in <%= and %>
  - file: '<%= pkg.name %>.zip'
- Predefine other properties of task as needed

Releasing to Github with Grunt

Custom Grunt tasks pull the repository's commit history

- Define the setreleasemsg task (Gruntfile.coffee, line 119)
  - Uses *$ git tag* to retrieve the last tag of a repository
  - Creates a new Grunt property named releaserange as lasttag..HEAD
- Define the shortlog task (Gruntfile.coffee, line 133)
  - Uses *$ git shortlog ___ --no-merges* with the releaserange property to retrieve a list of all commit messages since the last release, grouped by author
  - Hyphenates commit messages so Github shows it as an unordered list
  - Sets the gh_release task's body property with the finished release message

Releasing to Github with Grunt

1.1.8

a37fc58

Release

ZachWatkins released this on Oct 13, 2016 · **40 commits** to master since this release

Zachary Watkins (2):

- Restore list style appearance to menu widgets
- Add menu widget styles to other variations

Downloads

AgriFlex3.zip	1.06 MB
Source code (zip)	
Source code (tar.gz)	

New workflow

References and links