Setting Up Continuous Deployment from Github to WP Engine

Zachary Watkins
About me

• WordPress Developer for AgriLife Communications
• JavaScript, PHP, front end, and dev ops
• Written code for thousands of sites
What is Continuous Deployment?

• A workflow that minimizes the amount of time needed to provide users with the latest version of a code base
• Frequent commits to version control
• Automated tests, deployment, and feedback
Why should I use it?

• Easier to upload improvements and bug fixes - everywhere
• Easier to maintain testing and production builds of our code
• Easier to release an end-user package of our themes and plugins
• Faster deployment and troubleshooting
• Reduced dependency on FTP uploads
How long does it take to deploy one line change in your code?
Scenario

Update a link across 7 repositories, which are on either 6 or 8 servers each
A slow, difficult deployment workflow can delay your own innovation in favor of addressing user requests.
Steps to deploy a change

1. Make a change locally
2. Run staging build tasks
3. FTP in to staging servers
4. Upload the file to the right directory
5. If ready, run steps 1-5 for production
6. Push changes to Github repo
7. Ask users who clone our repo to run build scripts before using the plugin or theme
A faster, automated deployment will unburden this process and may allow time for innovation.
Steps with Continuous Deployment

1. Make a change locally
2. Push to repo’s staging branch
3. Test on staging server when Codeship finishes
4. Push to master branch
5. Double-check on production when deployed
6. Let users download the latest release as a .zip
Scenario

Update a link across 7 repositories, which are on either 6 or 8 servers each

5 minutes of work, 30 minutes of waiting for Codeship to push the changes to our servers
How did we accomplish this?
Basically, we…

1. Found a service compatible with web host
2. Learned how to connect it between our repos and servers
3. Wrote the code needed to facilitate this connection, and for making Github releases
Finding a service

• Searches for automated deployment to WP Engine pointed to their Git Push feature
• Articles affirmed compatibility between our host and Codeship
• Codeship’s free option allowed me to confirm
Learning the connecting points

- Git Push feature of WP Engine
- Codeship pipeline configuration, with environmental variables
- A reliable Grunt module for Github releases
Writing the code

• Bash commands for deploying on Codeship
• Codeship-only .gitignore file
• Grunt tasks to format and deploy Github releases
Example: AgriFlex3 Theme
AgriFlex3

• CSS built with Sass, Compass, and Foundation
• Different CSS build between environments
• On 6 of our servers
• Used outside of our team
Connecting Github and Codeship
Choose Github as the repository host

If you need a specific setup, please get in touch with us at sales@codeship.com. We would also love to get feedback from you.
Provide AgriFlex3’s clone URL

Connect Your GitHub Repository
Make sure you have access to the desired repository.

Connect your repository by providing the corresponding Git clone URL. Learn more about clone URLs

Examples:
» git@github.com:<username>/<repository_name>.git
» https://github.com:<username>/<repository_name>.git
» https://github.com/codeship/<repository_name>

Repository Clone URL
https://github.com/agrilife/agriflex3

Connect
Set up pipelines for your branches
Set up pipelines for your branches

Configure Your Deployment Pipelines

Create deployment pipelines per branch that will be executed after your tests passed. You are able to add multiple deployments per pipeline that will run sequentially.

- master
- staging

Configure a branch that triggers the deployment pipeline:

- Branch is exactly
- Branch Is exactly
- Branch starts with

Save pipeline settings
Building on Codeship
Building on Codeship

• Separate deployment pipelines for staging and production branches
• Set up repository info
• Add cli modules
• Change folder structure
• Apply Codeship-only gitignore
• Add repo dependencies
• Run build tasks
• Commit and push
Configure deployment pipelines

• Git shallow clone issue

```bash
# Get around shallow update restriction
if [ -f ${HOME}/clone/.git/shallow ]; then git fetch --unshallow; fi
```
Configure deployment pipelines

- Add user and remotes

```bash
# Add User Data
git config --global user.name "codeship-agriflex3"
git config --global user.email "zachary.watkins@ag.tamu.edu"
# Add servers
git remote add productionservers $AGRILIFE_PRODUCTION
git remote set-url --add --push productionservers $AGRILIFE_PRODUCTION
git remote set-url --add --push productionservers $COALS_PRODUCTION
git remote set-url --add --push productionservers $COUNTIES_PRODUCTION
git remote set-url --add --push productionservers $AGRILIFEEXTENSION_PRODUCTION
git remote set-url --add --push productionservers $TXMG_PRODUCTION
git remote set-url --add --push productionservers $TXMN_PRODUCTION
```
Configure deployment pipelines

• Add cli modules

```bash
# Install needed modules
npm install -g grunt-cli
npm install -g grunt@0.4.0
npm install -g bower
npm install -g ruby
gem install compass
```
Configure deployment pipelines

• Change folder structure

```bash
# Move repo files to a named folder
mkdir $FOLDERNAME
shopt -s extglob
mv !($FOLDERNAME) $FOLDERNAME

# Move repo files whose name begins with a period
mv .sass-lint.yml $FOLDERNAME/.sass-lint.yml

# Exclude development-only files from commit
rm .gitignore
mv .codeshipignore $FOLDERNAME/.gitignore

# Move named folder into a structure identical to the root directory of a WordPress server
mkdir -p $DIRECTORY
mv $FOLDERNAME $DIRECTORY
cd $DIRECTORY/$FOLDERNAME/
```
.codeshipignore

# Directories that may or may not exist in repo AND should not be on the server
package.json
bower.json
.bower.json
.bowerrc
config.rb
node_modules
.sass-cache
*.md
*.txt
*.ai
*.scss
*.coffee
.gitignore
LICENSE
LICENSE-MIT
gruntfile.js
Gruntfile.js
werker.yml
.editorconfig

bower_components/**/foundation/scss/
bower_components/**/jquery/src/
bower_components/**/modernizr/**/
bower_components/**/modernizr/grunt.js
bower_components/**/picturefill/index.html
bower_components/**/picturefill/logos/**.png
bower_components/**/superfish/examples/

# Only using one file from bower_components/html5shiv
bower_components/html5shiv/
!bower_components/html5shiv/dist/html5shiv.js

# Prevent Composer files from being loaded if they are only used for PHP5.6+
vendor/composer/autoload_static.php

# Ignore release file
AgriFlex3.zip
Configure deployment pipelines

- Add repo dependencies, run build tasks

```bash
# Build
composer install
npm install
bower install html5shiv#3.7.0
bower install respond#1.4.1
grunt package
```
Configure deployment pipelines

- Commit, push to servers, release to Github

```bash
# Deploy
git add --all .
git commit -m "DEPLOYMENT"
git push productionservers HEAD:master --force
grunt release
```
Add environmental variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXMN_PRODUCTION</td>
<td><a href="mailto:git@git.wpengine.com">git@git.wpengine.com</a>:production/installname.git</td>
</tr>
<tr>
<td>TXMN_STAGING</td>
<td><a href="mailto:git@git.wpengine.com">git@git.wpengine.com</a>:staging/installname.git</td>
</tr>
<tr>
<td>FOLDERNAME</td>
<td>AgriFlex3</td>
</tr>
<tr>
<td>DIRECTORY</td>
<td>wp-content/themes</td>
</tr>
<tr>
<td>RELEASE_KEY</td>
<td>abc123abc123abc123abc123abc123abc123abc123abc123</td>
</tr>
</tbody>
</table>

Add a new environment variable

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>VALUE</td>
</tr>
</tbody>
</table>

Save Configuration
Connecting Codeship and WP Engine
Set a notification for Slack

Slack
To get started add our Codeship integration to your team's Slack account.

Enable: 

Webhook URL: https://hooks.slack.com/services/manyhashes
Git Push User

- Copy Codeship project’s SSH public key
Git Push User

- Add WP Engine Git Push user with that key
Releasing to Github with Grunt
Releasing to Github with Grunt

• Compress target files
• Custom Grunt tasks for release message
• Define an environmental variable using a Github personal access token
• Grunt module for making Github releases
Example Release

Release

Zach Watkins released this 6 days ago · 6 commits to master since this release

Zachary Watkins (3):

- Improve release message scripts for code readability
- Applying sass-lint ignore statements for the no-important rule
- Update State Link Policy URL

Downloads

- AgriFlex3.zip
- Source code (zip)
- Source code (tar.gz)
Target files for compression

• grunt-contrib-compress module
• File paths may use Grunt’s globbing patterns

```javascript
compress:  
  main:  
    options:  
      archive: 'AgriFlex3.zip'  
    files: [  
      {src: ['AgriFlex/*.php']},  
      {src: ['css/*.css']},  
      {src: ['img/**']},  
      {src: ['js/*.js']},  
      {src: ['bower_components/fastclick/lib/fastclick.js']},  
      {src: ['bower_components/foundation/{css,js}/**']},  
      {src: ['bower_components/modernizr/modernizr.js']},  
      {src: ['bower_components/jquery/{dist,sizzle}/**/*.js']},  
      {src: ['bower_components/jquery-placeholder/*.js']},  
      {src: ['bower_components/jquery.cookie/jquery.cookie.js']},  
      {src: ['bower_components/respond/{cross-domain,dest}/*.js']},  
      {src: ['bower_components/html5shiv/dist/html5shiv.js']},  
      {src: ['vendor/**', '!vendor/composer/autoload_static.php']},  
      {src: ['functions.php']},  
      {src: ['README.md']},  
      {src: ['rtl.css']},  
      {src: ['screenshot.png']},  
      {src: ['search.php']},  
      {src: ['style.css']}
    ]
```
Custom Grunt tasks for release message

- The “setreleasemsg” task gets the last release’s tag, then runs the shortlog task.
Custom Grunt tasks for release message

• The “shortlog” task adds the formatted release message

```coffee
@registerTask 'shortlog', 'Set gh_release body with commit messages since last release', ->
  done = @async()
  releaserange = grunt.template.process '<%= lasttag %>..HEAD'
  grunt.util.spawn {
    cmd: 'git'
    args: ['shortlog', releaserange, '--no-merges']
  }, (err, result, code) ->
  if result.stdout isnt ''
    message = result.stdout.replace /(\n)\s+/g, '$1-
    message = message.replace /\s\*[skip ci]/g, ''
    grunt.config 'gh_release.release.body', message
  done(err)
return
return
```
Github personal access token variable

• Use [https://github.com/settings/tokens](https://github.com/settings/tokens) to generate a new token scoped to public repos
Github personal access token variable

- Use [https://github.com/settings/tokens](https://github.com/settings/tokens) to generate a new token scoped to public repos.
Github personal access token variable

• Define it as an environmental variable named RELEASE_KEY - Mac
Github personal access token variable

- Define it as an environmental variable named RELEASE_KEY - Mac

```bash
zacharywatkins$ cd && nano .bash_profile

export RELEASE_KEY="yournewpersonalaccesstoken"
export NVM_DIR="/Users/zacharywatkins/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"  # This loads nvm
```
Github personal access token variable

- Define it as an environmental variable named RELEASE_KEY - Mac

```bash
export RELEASE_KEY="yournewpersonalaccesstoken"
export NVM_DIR="/Users/zacharywatkins/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh" # This loads nvm
```

```bash
Zacharys-MacBook-Pro:~ zacharywatkins$ cd ~ && nano .bash_profile
```

```bash
Zacharys-MacBook-Pro:~ zacharywatkins$ source ~/.bash_profile
```
Github personal access token variable

• Define it as an environmental variable named RELEASE_KEY – Windows
• Go to Control Panel -> System -> Environment and define it with the GUI
Grunt module for Github releases

- grunt-gh-release module
- requires the access token variable

```coffee
gh_release:
  options:
    token: process.env.RELEASE_KEY
    owner: 'agrilife'
    repo: '<%= pkg.name %>'
  release:
    tag_name: '<%= pkg.version %>'
    target_commitish: 'master'
    name: 'Release'
    body: 'Release'
    draft: false
    prerelease: false
  asset:
    name: '<%= pkg.name %>.zip'
    file: '<%= pkg.name %>.zip'
    'Content-Type': 'application/zip'
```
What are the benefits?
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
• Faster deployment and troubleshooting
• Reduced dependency on FTP uploads
Worth noting

• WP Engine checks PHP files you push for errors and incompatibility with the version of PHP that is on the target server

• Codeship has commit message keywords that allow you to avoid building a commit
Credits

• **Continuous Deployment**, agilealliance.org
• Alex Zuniga (WP Engine) for pointing me to an **improved shallow push solution**
Resources

• AgriFlex3 Github repository
• Principles and Practices in Continuous Deployment at Etsy, by Mike Brittain
• grunt-contrib-compress
• grunt-gh-release
• node-glob
• Continuously Deploy your Wordpress Theme to WPEngine with Codeship, by Roman Kuba