Automating WordPress Development

Chris Wiegman
https://chriswiegman.com | @ChrisWiegman
http://wieg.co/wcsea19

Find the slides at

http://wieg.co/wcsea19

About Me

- **Senior Software Engineer –** WP Engine
- **iThemes Security** (Better WP Security)
- St. Edward’s University
- Privacy
- Development Workflows
- Aviation

“Automation is good, so long as you know exactly where to put the machine.”

- Eliyahu Goldratt

There are many machines for WP

1) Downloading the site (or installing a fresh copy of WordPress)
2) Developing the site/theme/plugin
3) Testing the project
4) Debugging the project
5) Presenting the project for stakeholder review
6) Deploying the project

Download Existing Work

Download the Site

1) Setup local server
2) Log into remote server
3) Copy files from remote to local
4) Log into database
5) Export database/import locally
6) Search/replace domains
7) Profit???

Use a Modern Tool

- WP Engine DevKit or Local Lightning
- Your host’s solution
- Bash (or similar) script

- 1-click setup
- Reduce external connections
- Reduce your stress level

Downloading Plugins/Themes

Use version control (Git)

Starting New Code

Creating a new Plugin

```
<?php
/**
 * My Awesome Plugin
 * @version 1.0
 **/

add_action( ‘init’, ‘hello’, 1 );

function hello() {
    wp_die( ‘Hi Roy’ );
}
```

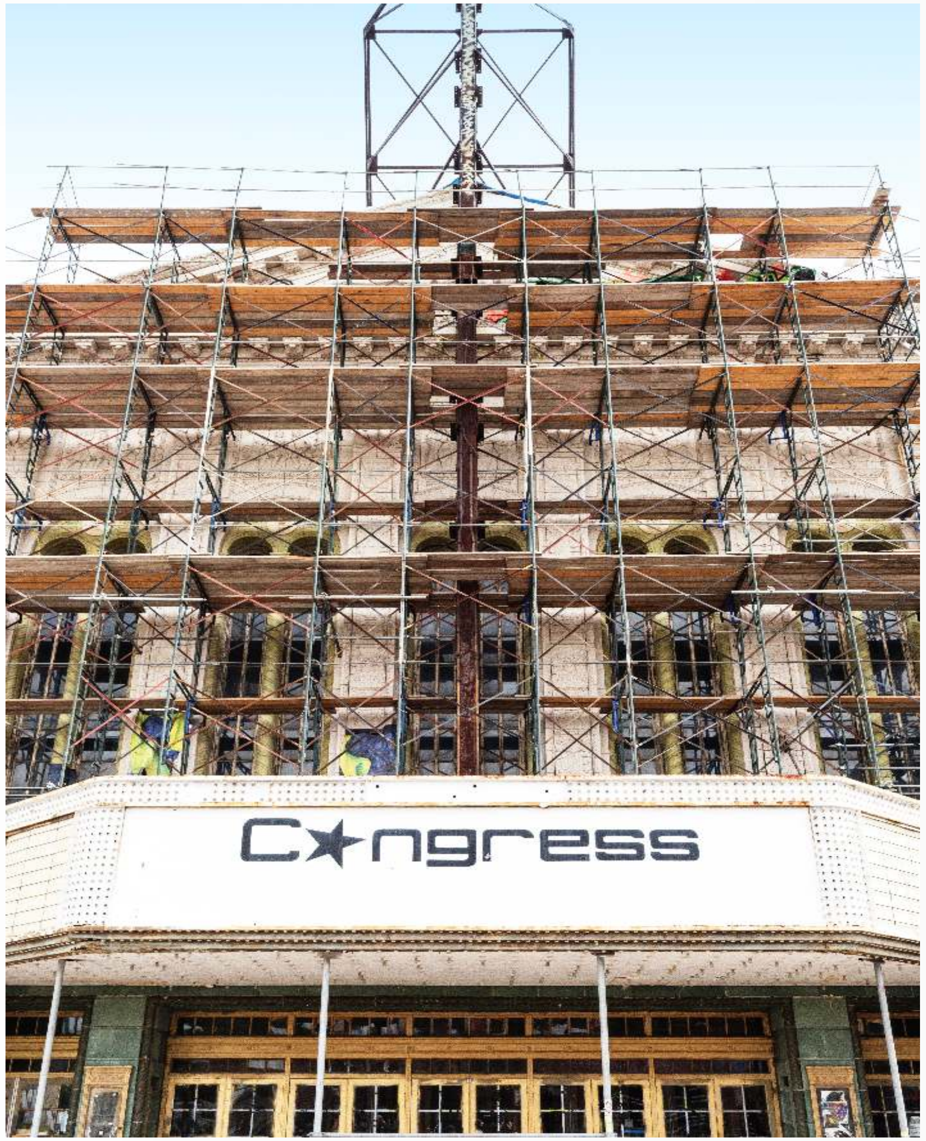
Where does the code go?

What files should I create?

What if I want [SASS/Webpack/etc]?

Code Scaffolding

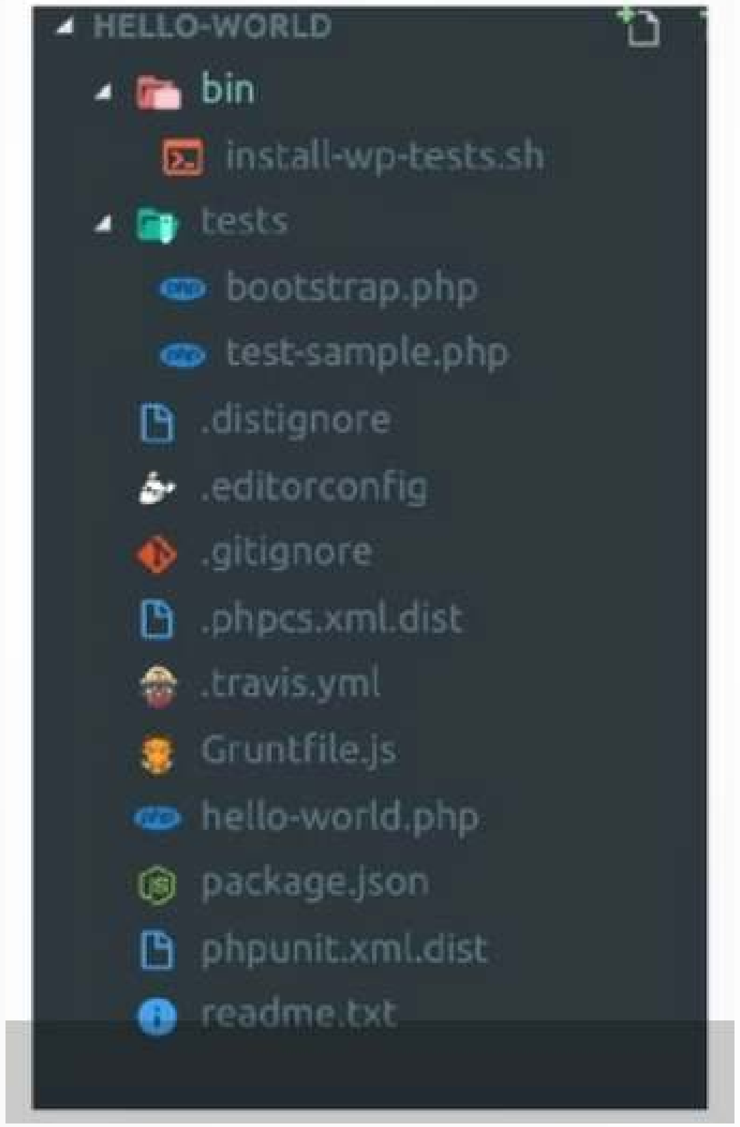
- Easily reproducible
- Enforce best practices
- Opinionated
- Testing built-in
- Build tools already configured



wp-cli Scaffold

- *wp scaffold* is built into wp-cli
- Can build:
  - Plugins
  - Themes (child theme or based on _s)
  - Blocks
  - Plugin tests
  - Theme tests
  - *And more (https://developer.wordpress.org/cli/commands/scaffold/)*

Creating a Plugin



```
HELLO-WORLD
  bin
    install-wp-tests.sh
  tests
    bootstrap.php
    test-sample.php
  .distignore
  .editorconfig
  .gitignore
  .phpcs.xml.dist
  .travis.yml
  Gruntfile.js
  hello-world.php
  package.json
  phpunit.xml.dist
  readme.txt
```

wp scaffold plugin hello-world

- Includes:
  - Basic plugin file
  - .gitignore
  - Travis
  - Grunt
  - Unit tests
  - Editor config

wp-cli scaffold: Not Just Full Projects

wp scaffold post-type movie –label=Movie –plugin=hello-world

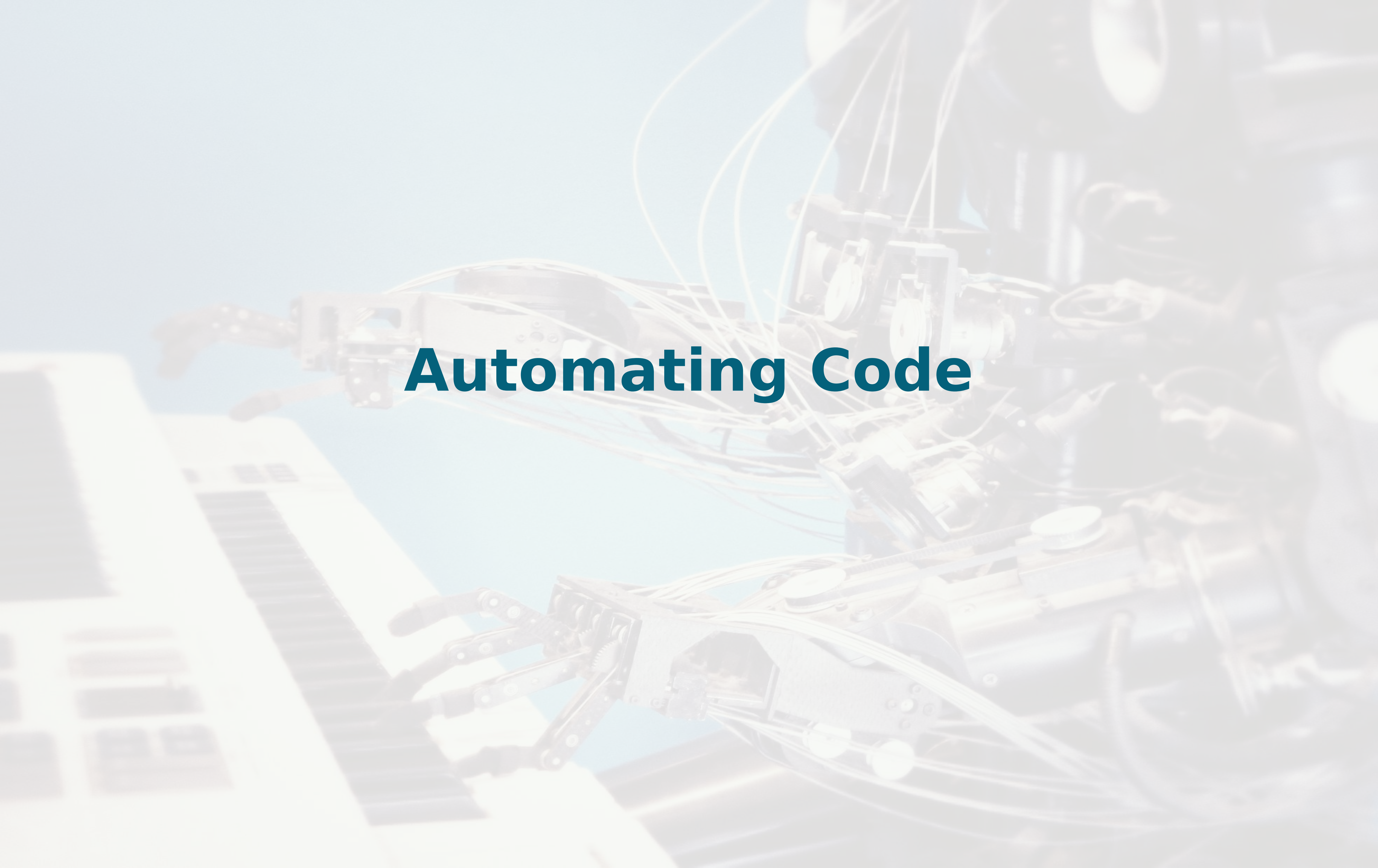
- Create code for a “movie” custom post type in the hello-world plugin

When wp-cli scaffold Doesn’t Cut It

- Problems with wp-cli scaffold
  - It’s opinionated
  - Is Grunt still a “thing?”
    (it did restart some development in June 2018)
  - Complex file structures don’t exist
  - Themes only use _s (underscores)

Alternatives to wp-cli scaffold

- Write your own wp-cli scaffold sub-command
- Yeoman
  - Generator WP (https://gitea.chriswiegman.com/chriswiegman/generator-wp)
  - Generator WP Make (https://github.com/10up/generator-wp-make)
  - Role your own
    - GoLang
    - PHP
    - JavaScript
    - *etc*



Automating Code

Syntax Doesn't Matter

- WP coding standards set the standards for code syntax
- PHP_CodeSniffer
  - Tells you when you differ from WP coding standards
    - Performance
    - Security
    - Syntax
  - Phpcbd (or editor's alternative)
    - Automagically fix syntax errors in your code
    - Spaces, tabs and more no longer matter

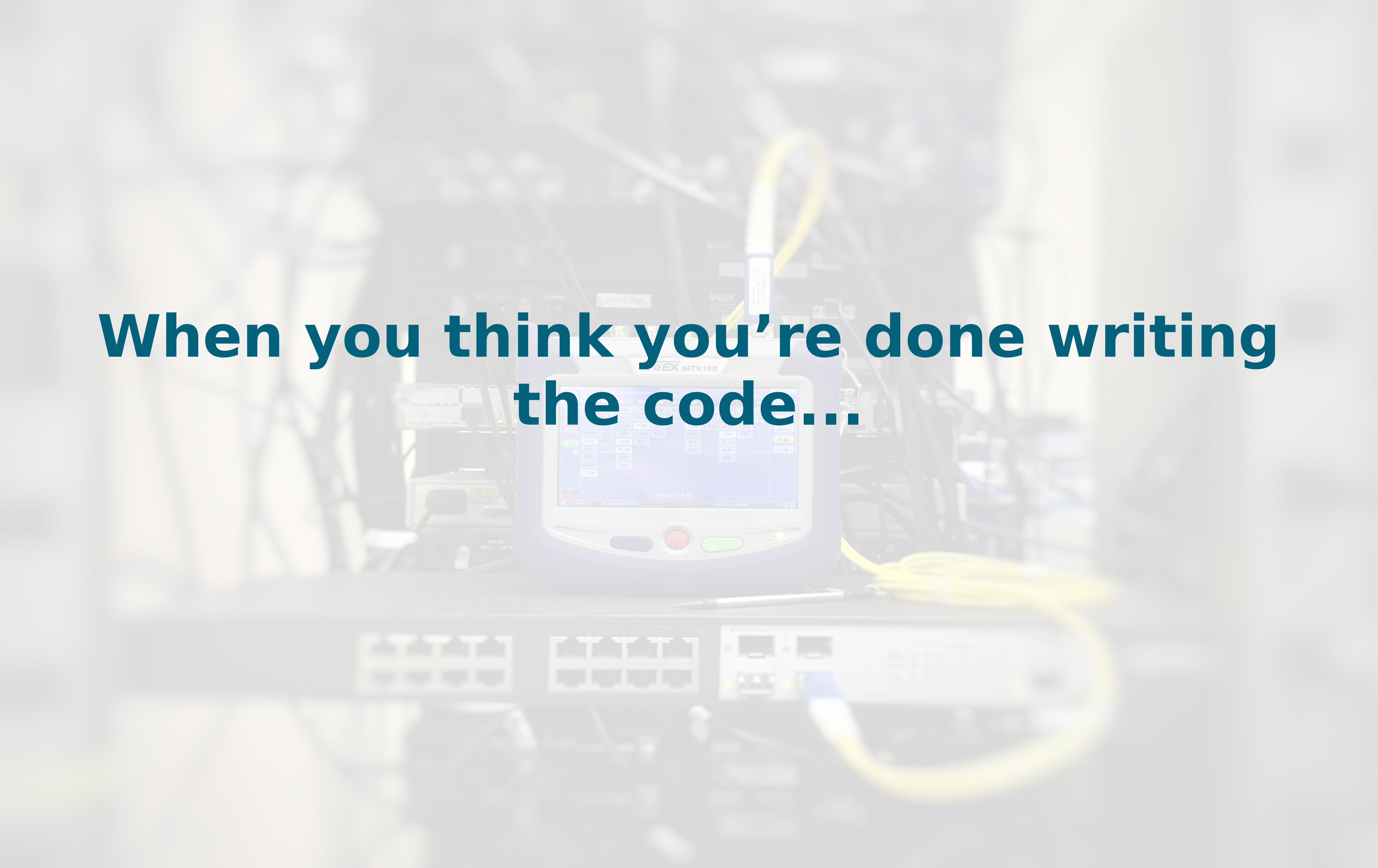
Finding Bugs

- Step-through debugging helps automate searching for bugs in code
  - No more console.log() or var_dump() statements
  - JavaScript
    - Look for your browser/editor combination
  - PHP – Xdebug
    - Works with all browsers and major editors
    - See all variables where they occur, step back until problem occurs
    - Profile page load to find deeper issues (simple alternative to New Relic)

Task runners for the rest

- Grunt/Gulp/NPM/Webpack/etc to handle misc tasks
  - Minimize JS
  - Process/Minimize SASS/CSS/etc
  - Optimize images
  - Create i18n (translation) files

When you think you’re done writing the code…



Enforcing standards and more

- Just like WordPress, Git offers hooks
- Pre-commit hooks must succeed for a commit to continue
- WP_Enforcer (https://github.com/stevegrunwell/wp-enforcer)
- Could include build assets if added to repository
  - Build assets probable shouldn’t be added to your repository

What more testing do we need?

- Xdebug and PHP_CodeSniffer are great while writing code
  - Don’t do much for you later
- WP-cli scaffold gave us a phpunit framework…
  - Which does little if we don’t use it
- Does your code break anything else in WordPress?
- Has every developer setup tools such as PHP_Codesniffer

Enter CI/CD

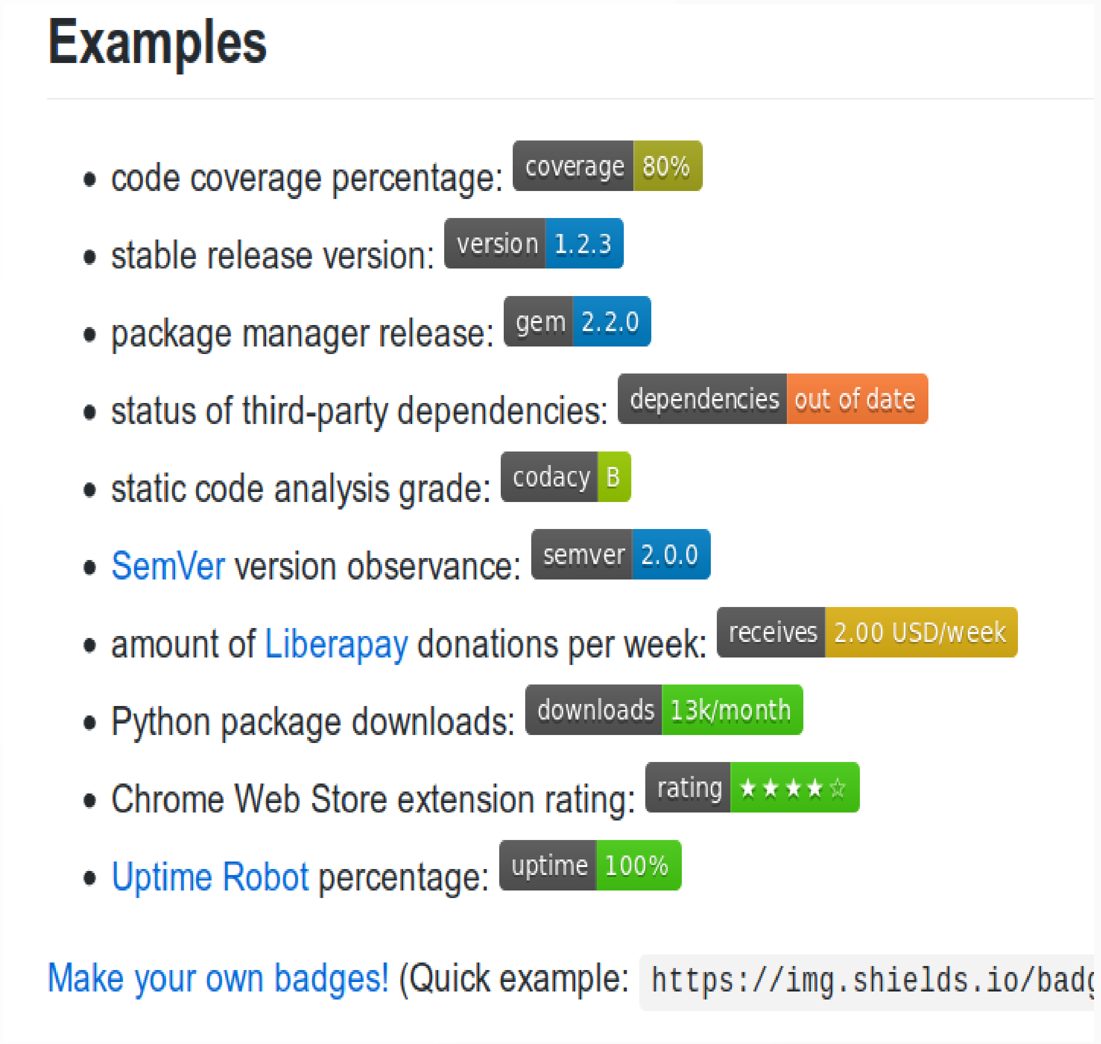
- Continuous Integration
- Continuous Delivery/Deployment
- Probably built into your Git host
  - GitHub – Travis
  - GitLab – GitLab CI
  - Jenkins, Circle CI, *many more*
- Three steps to CI/CD
  - Build, Test, Deploy

The build step

- Execute the tasks in your task runner
  - Build all project assets (CSS/JS/i18n/etc)
- Setup for any testing
- At the end of the build step you should have a package that *could* be given to an end user

The test step

- Run unit, integration, acceptance and any other testing
  - WP Acceptance
  - Jest (or other framework)
- Computer phpunit or other test coverage
- Fail if there are any issues



Examples

- code coverage percentage: coverage 80%
- stable release version: version 1.2.3
- package manager release: gem 2.2.0
- status of third-party dependencies: dependencies out of date
- static code analysis grade: codacy B
- SemVer version observance: semver 2.0.0
- amount of Liberapay donations per week: receives 2.00 USD/week
- Python package downloads: downloads 13k/month
- Chrome Web Store extension rating: rating ★★★★☆
- Uptime Robot percentage: uptime 100%

Make your own badges! (Quick example: `https://img.shields.io/badg`

Deploying Your Code

Using CI/CD

- Version your project
- Copy files
- Trigger remote Git pull
- Run a deployment script

Deploying to WordPress.org

- Bash can handle it all
  - Example: (https://github.com/aaroneaton/better-yourls/blob/master/deploy.sh)
    - Checks plugin version
    - Handles all SVN commits and tagging on WordPress.org
    - Can work for themes or plugins
    - *Do **NOT** use it on your first submission*

What about the changelog?

- Follow your progress with Conventional Commits
  - https://www.conventionalcommits.org/
  - Examples:
    - fix(post types): Fixed the post type bug
    - feat(blocks): Added a new block
  - Process with Conventional Commits CLI
    - https://www.npmjs.com/package/conventional-changelog-cli
  - Often best done in the deploy process

Latest release

v0.14.4

2cd613f

v0.14.4

octalmage released this 7 days ago · 83 commits to master since this release

Edit

0.14.4 (2019-08-14)

Bug Fixes

- Pass SSHKey to SSH when pulling the database. Solves related 255 errors. (368f2b5)
- **clone:** Don't leave an empty site directory when a clone bails for multisite (ee41ece)
- Capture more logs on failed startup to help debug. (019d0d1)

Features

- **setup:** Improve SSH workflow in setup prompts (4d2d1cd)
- Improve SSH guidance after clone failure. (5a0f1b6)

Assets 8

v0.14.3

cefcf2b

v0.14.3

octalmage released this 12 days ago · 121 commits to master since this release

Edit

Combining complex workflows

- Make (https://www.gnu.org/software/make/)
  - Designed for files, but can do so much more
    - *make build-assets*
    - *make test-unit*
    - *make test-acceptance*
    - *make release-changelog*
    - *make release-deploy*

An example make task

```
release-changelog:
    @echo "Generating the changelog and adding it to the
release."
    rm -f $(CHANGELOG_FILE)
    $(DOCKER_UTILITY_CMD) npx conventional-changelog-cli \
    -s \
    -p angular \
    -i $(CHANGELOG_FILE) \
    -r $(RELEASES) \
    -n ./.changelog-options.js
```

Pitfalls of Automation

Automation doesn’t solve your problems.

The ROI of automation is realized over time.

One size does not fit all.

Not every process needs automation.

Questions

Thank you!

Slides: http://wieg.co/wcsea19

https://chriswiegman.com | @Chriswiegman