Deploying Drupal with Fabric

Oliver Davies

bit.ly/deploying-drupal-fabric

- What is Fabric and what do I use it for?
- How to write and organise Fabric scripts
- Task examples

- Senior Developer at Microserve
- Part-time freelance Developer & System Administrator
- Drupal, Symfony, Silex, Laravel, Sculpin
- Drupal Bristol, PHPSW, DrupalCamp Bristol
- Sticker collector, herder of elePHPants
- @opdavies
- oliverdavies.uk

What is Fabric?

What is Fabric?

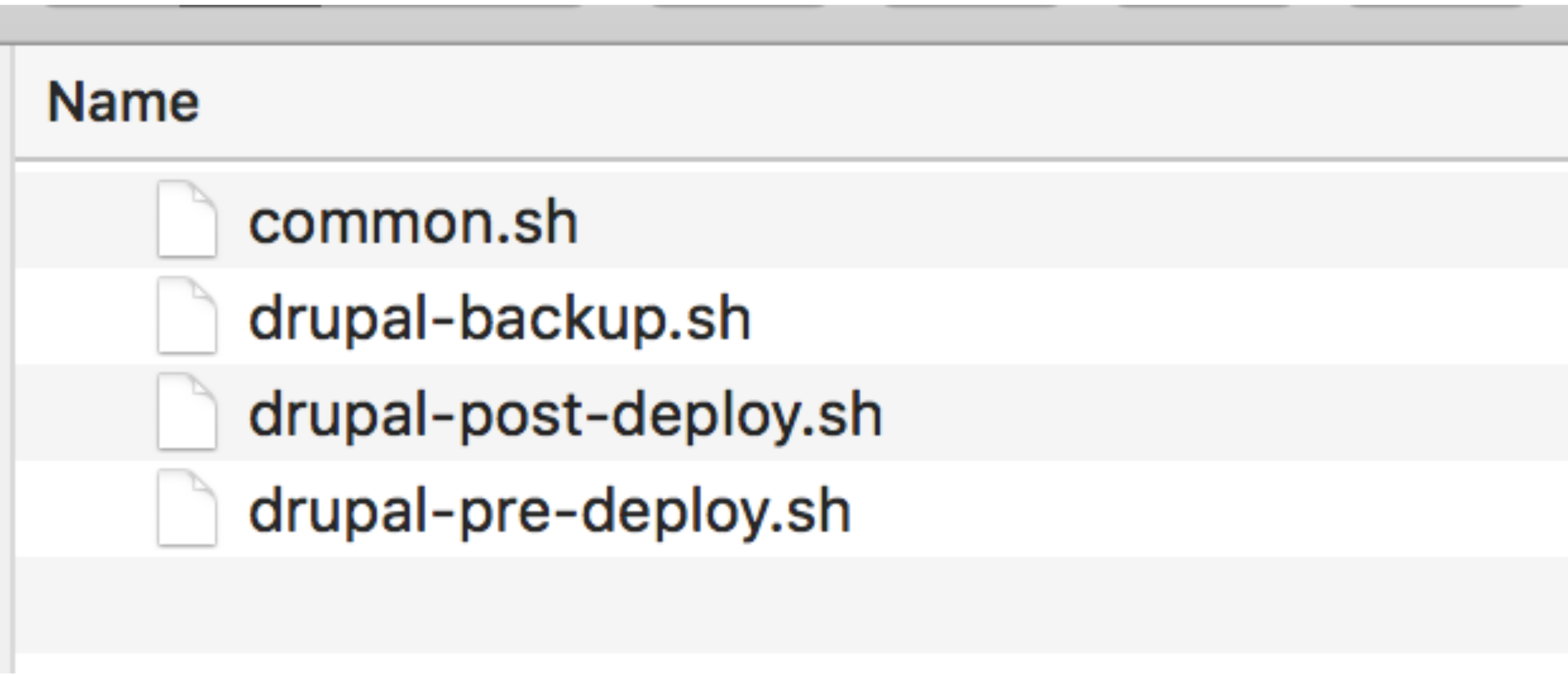
Fabric is a Python (2.5-2.7) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks.

What is Fabric?

It provides a basic suite of operations for executing local or remote shell commands (normally or via sudo) and uploading/downloading files, as well as auxiliary functionality such as prompting the running user for input, or aborting execution.

I use Fabric to…

- Simplify my build process
- Deploy code directly to different environments
- Act as an intermediate step



Name

common.sh
drupal-backup.sh
drupal-post-deploy.sh
drupal-pre-deploy.sh

Why Fabric?

- Powerful
- Flexible
- Easier to read and write than bash

Installing Fabric

```
$ pip install fabric

# macOS
$ brew install fabric

# Debian, Ubuntu
$ apt-get install fabric
$ apt-get install python-fabric
```

Writing your first fabfile

```python
# fabfile.py

from fabric.api import env, run, cd, local

env.hosts = ['example.com']

# Do stuff...
# fabfile.py

from fabric.api import *

env.hosts = ['example.com']

# Do stuff...
```

Operations

- cd, lcd - *change directory*
- run, sudo, local - *run a command*
- get - *download files*
- put - *upload files*

http://docs.fabfile.org/en/1.13/api/core/operations.html

Utils

- warn: *print warning message*
- abort: *abort execution, exit with error status*
- error: *call func with given error message*
- puts: *alias for print whose output is managed by Fabric's output controls*

http://docs.fabfile.org/en/1.13/api/core/utils.html

File management

```
from fabric.contrib.files import *
```

- exists - *check if path exists*
- contains - *check if file contains text/matches regex*
- sed - *run search and replace on a file*
- upload_template - *render and upload a template to remote host*

http://docs.fabfile.org/en/1.13/api/contrib/files.html#fabric.contrib.files.append

Tasks

```python
def main():
  with cd('/var/www/html'):
    run('git pull')
    run('composer install')
```

Task arguments

```
def main(run_composer=True):
  with cd('/var/www/html'):
    run('git pull')

    if run_composer:
      run('composer install')
```

Task arguments

```
def main(run_composer=True, env='prod', build_type):
  with cd('/var/www/html'):
    run('git pull')

    if run_composer:
      if env == 'prod':
        run('composer install --no-dev')
      else:
        run('composer install')

      if build_type == 'drupal':
        ...
      elif build_type == 'symfony':
        ...
      elif build_type == 'sculpin':
        ...
```

Calling other tasks

```
@task
def main():
  with cd('/var/www/html'):
    build()
    post_install()

def build():
  run('git pull')
  run('composer install')

def post_install():
  with prefix('drush'):
    run('updatedb -y')
    run('entity-updates -y')
    run('cache-rebuild')
```

Running Tasks

```
fab --list

fab <task>

fab <task>:build_number=$BUILD_ID,build_type=drupal
```

```
[production] Executing task 'main'
[production] run: git pull
[production] out: Already up-to-date.
[production] out:

[production] run: composer install
...
[production] out: Generating autoload files
[production] out:

Done.
Disconnecting from production... done.
```

Downsides

- Running build tasks on production

Not Building on Prod

1. Build locally and deploy.

Local tasks

```
# Runs remotely.

from fabric.api import run

run('git pull')
run('composer install')

# Runs locally.

from fabric.api import local

local('git pull')
local('composer install')
```

Local tasks

```
# Remote.

from fabric.api import cd

with cd('themes/custom/drupalbristol'):
  ...

# Runs locally.

from fabric.api import lcd

with lcd('themes/custom/drupalbristol'):
  ...
```

rsync

```
from fabric.contrib.project import rsync_project

...

def deploy():
  rsync_project(
    local_dir='./',
    remote_dir='/var/www/html'
    default_opts='-vzcrSLh',
    exclude=('.git', 'node_modules/', '.sass-cache/')
  )
```

```
[production] Executing task 'main'
[localhost] local: git pull
Current branch master is up to date.
[localhost] local: composer install
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Nothing to install or update
Generating autoload files

Done.
```

Not Building on Prod

1. ~~Build locally and deploy.~~
2. Build in a separate directory and switch after build.

Deploying into a different directory

```python
from fabric.api import *
from time import time

project_dir = '/var/www/html'
next_release = "%(time).0f" % { 'time': time() } # Current timestamp

def init():
  if not exists(project_dir):
    run('mkdir -p %s/backups' % project_dir)
    run('mkdir -p %s/shared' % project_dir)
    run('mkdir -p %s/releases' % project_dir)
```

Deploying into a different directory

```
current_release = '%s/%s' % (releases_dir, next_release)

run('git clone %s %s' % (git_repo, current_release))

def build():
  with cd(current_release):
    pre_tasks()
    build()
    post_tasks()
```

Deploying into a different directory

```
def pre_build(build_number):
  with cd('current'):
    print '==> Dumping the DB (just in case)...'
    backup_database()

def backup_database():
  cd('drush sql-dump --gzip > ../backups/%s.sql.gz' % build_number)
```

Deploying into a different directory

```
def update_symlinks():
  run('ln -nfs %s/releases/%s %s/current'
    % (project_dir, next_release, project_dir))

# /var/www/html/current
```

```
[production] Executing task 'main'
[production] run: git clone https://github.com/opdavies/oliverdavies.uk.git
  /var/www/html/releases/1505865600
Installing Composer dependencies...
[production] run: composer install --no-dev
Update the symlink to the new release...
[production] run: ln -nfs /var/www/html/releases/1505865600
  /var/www/html/current

Done.
```

```
# /var/www/html

shared # settings.local.php, sites.php, files etc.
releases/1502323200
releases/1505692800
releases/1505696400
releases/1505865600
current -> releases/1505865600 # symlink
```

Positives

- Errors happen away from production

Downsides

- Lots of release directories

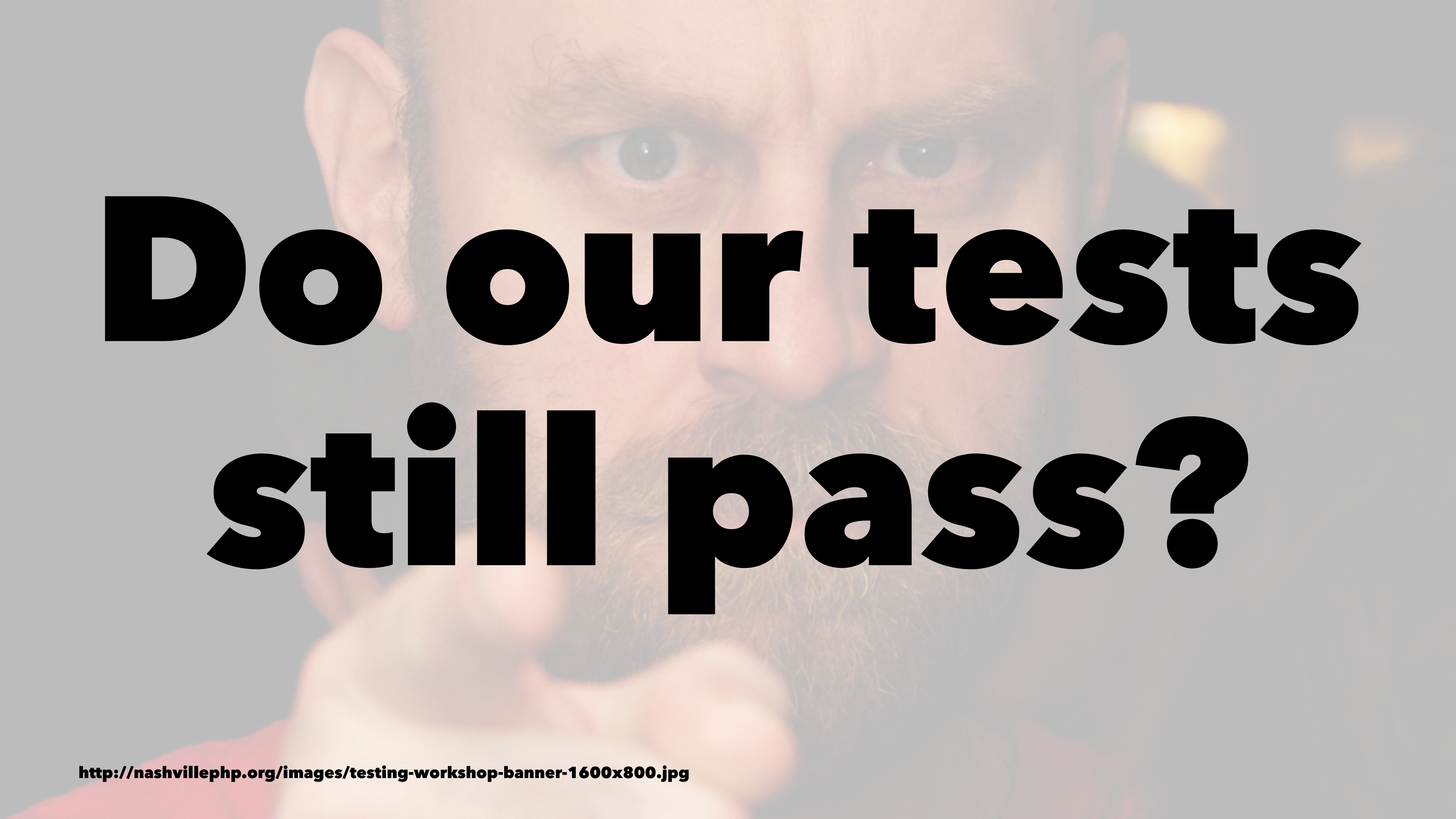
Removing old builds

```
def main(builds_to_keep=3):
  with cd('%s/releases' % project_dir):
    run("ls -1tr | head -n -%d | xargs -d '\\n' rm -fr"
      % builds_to_keep)
```

Does the code still merge cleanly?

```
def check_for_merge_conflicts(target_branch):
  with settings(warn_only=True):
    print('Ensuring that this can be merged into the main branch.')

    if local('git fetch && git merge --no-ff origin/%s'
      % target_branch).failed:
      abort('Cannot merge into target branch.')
```



Do our tests still pass?

http://nashvillephp.org/images/testing-workshop-banner-1600x800.jpg

```
with settings(warn_only=True):
  with lcd('%s/docroot/core' % project_dir):
    if local('../../vendor/bin/phpunit ../modules/custom').failed:
      abort('Tests failed!')
```

```
[localhost] run: ../../vendor/bin/phpunit ../modules/custom
[localhost] out: PHPUnit 4.8.35 by Sebastian Bergmann and contributors.
[localhost] out:
[localhost] out: .......
[localhost] out:
[localhost] out: Time: 1.59 minutes, Memory: 6.00MB
[localhost] out:
[localhost] out: OK (7 tests, 42 assertions)
[localhost] out:


Done.
```

```
[localhost] run: ../../vendor/bin/phpunit ../modules/custom
[localhost] out: PHPUnit 4.8.35 by Sebastian Bergmann and contributors.
[localhost] out:
[localhost] out: E
[localhost] out:
[localhost] out: Time: 18.67 seconds, Memory: 6.00MB
[localhost] out:
[localhost] out: There was 1 error:
[localhost] out:
[localhost] out: 1) Drupal\Tests\broadbean\Functional\AddJobTest::testNodesAreCreated
[localhost] out: Behat\Mink\Exception\ExpectationException: Current response status code is 200, but 201 expected.
[localhost] out:
[localhost] out: /var/www/html/vendor/behat/mink/src/WebAssert.php:770
[localhost] out: /var/www/html/vendor/behat/mink/src/WebAssert.php:130
[localhost] out: /var/www/html/docroot/modules/custom/broadbean/tests/src/Functional/AddJobTest.php:66
[localhost] out:
[localhost] out: FAILURES!
[localhost] out: Tests: 1, Assertions: 6, Errors: 1.
[localhost] out:

Warning: run() received nonzero return code 2 while executing '../../vendor/bin/phpunit ../modules/custom/broadbean'!

Fatal error: Tests failed!

Aborting.
```

Is the site still running?

Checking for failures

```
run(command).failed:
  # Fail

run(command).return_code == 1:
  # Fail

run(command).return_code == 0:
  # Pass
```

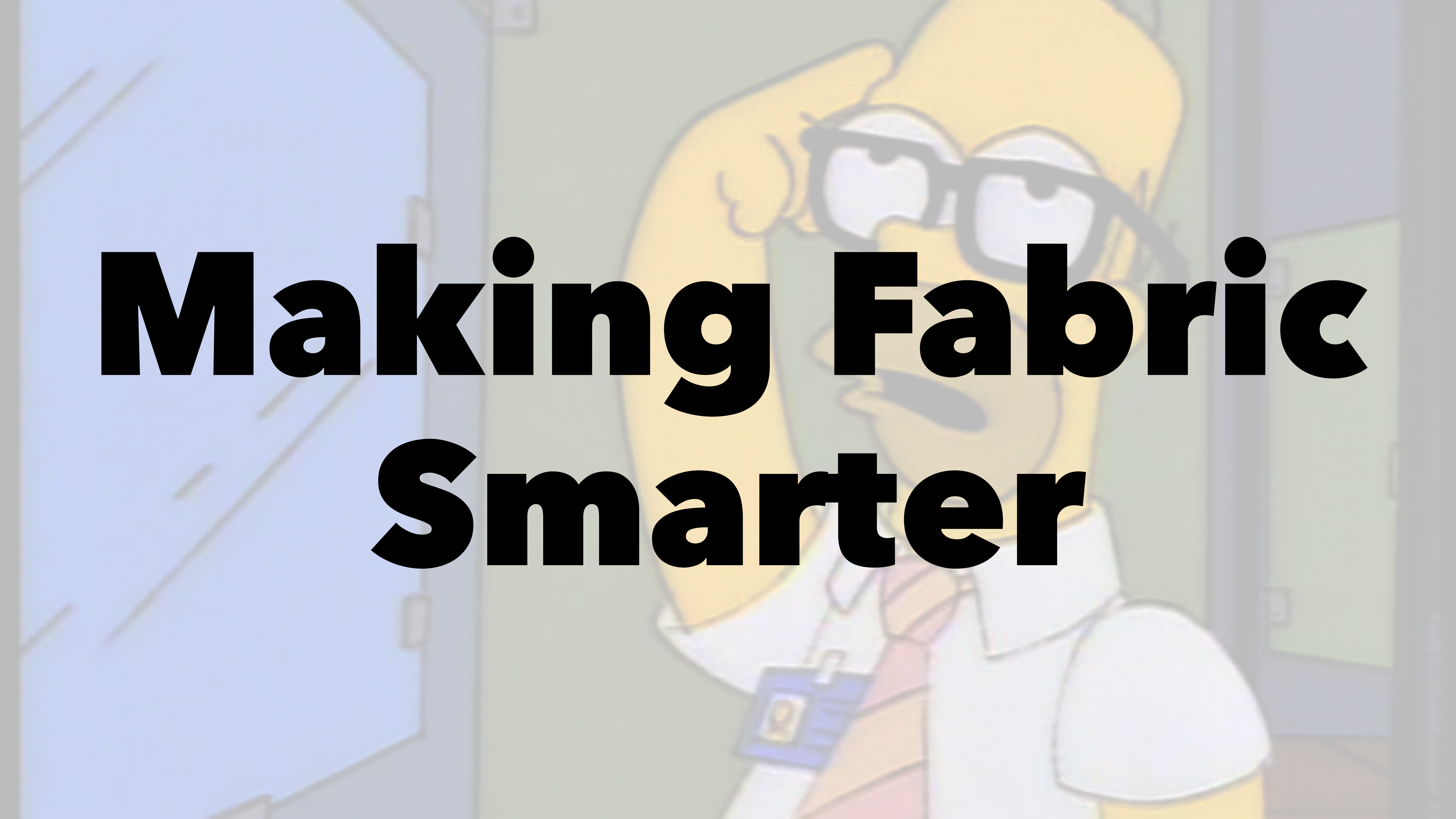
```
print 'Checking the site is alive...'
if run('drush status | egrep "Connected|Successful"').failed:
  # Revert back to previous build.
```

```
$ drush status

Drupal version                      :  8.3.7
Site URI                            :  http://default
Database driver                     :  mysql
Database hostname                   :  db
Database username                   :  user
Database name                       :  default
Database                            :  Connected
Drupal bootstrap                    :  Successful
Drupal user                         :
Default theme                       :  bartik
Administration theme                :  seven
PHP configuration                   :  /etc/php5/cli/php.ini
...
```

```
$ drush status

Drupal version                    :  8.3.7
Site URI                          :  http://default
Database driver                   :  mysql
Database hostname                 :  db
Database username                 :  user
Database name                     :  default
PHP configuration                 :  /etc/php5/cli/php.ini
...
```



Making Fabric Smarter

Conditional variables

```
drupal_version = None

if exists('composer.json') and exists('core'):
  drupal_version = 8
else:
  drupal_version = 7
```

Conditional tasks

```
if exists('composer.json'):
  run('composer install')

with cd('themes/custom/example'):
  if exists('package.json') and not exists('node_modules'):
    run('yarn --pure-lockfile')

  if exists('gulpfile.js'):
    run('node_modules/.bin/gulp --production')
  elif exists('gruntfile.js'):
    run('node_modules/.bin/grunt build')
```

Project settings file

```yaml
# app.yml

drupal:
  version: 8
  root: web
  config:
    import: yes
    name: sync
    cmi_tools: no
  tests:
    simpletest: false
    phpunit: true
  theme:
    path: 'themes/custom/drupalbristol'
    build:
      type: gulp
      npm: no
      yarn: yes

composer:
  install: true
```

Project settings file

```
# fabfile.py

from fabric.api import *
import yaml

config = []

if exists('app.yml'):
  with open('app.yml', 'r') as file:
    config = yaml.load(file.read())
```

Project settings file

```
# fabfile.py

if config['composer']['install'] == True:
  local('composer install')
```

Project settings file

```
# fabfile.py

if build_type == 'drupal':
  drupal = config['drupal']
```

Project settings file

```
# fabfile.py

if build_type == 'drupal':
  drupal = config['drupal']

  with cd(drupal['root']):
    if drupal['version'] == 8:

    if drupal['version'] == 7:
```

Project settings file

```
# fabfile.py

if build_type == 'drupal':
  drupal = config['drupal']

  with cd(drupal['root']):
    if drupal['version'] == 8:
      if drupal['config']['import'] == True:
        # Import the staged configuration.
        run('drush cim -y %s' % drupal['config']['name'])
```

Project settings file

```
# fabfile.py

if build_type == 'drupal':
  drupal = config['drupal']

  with cd(drupal['root']):
    if drupal['version'] == 8:
      if drupal['config']['import'] == True:
        if drupal['config']['cmi_tools'] == True:
          # Use Drush CMI Tools.
          run('drush cimy -y %s' % drupal['config']['name'])
        else:
          # Use core.
          run('drush cim -y %s' % drupal['config']['name'])
```

Project settings file

```
# fabfile.py

theme = config['theme']

with cd(theme['path']):
  if theme['build']['gulp'] == True:
    if env == 'prod':
      run('node_modules/.bin/gulp --production')
    else:
      run('node_modules/.bin/gulp')
```

Project settings file v2

```
# app.yml

commands:
  build: |
    cd web/themes/custom/drupalbristol
    yarn --pure-lockfile
    node_modules/.bin/gulp --production

  deploy: |
    cd web
    drush cache-rebuild -y
```

Project settings file v2

```
# fabfile.py

for hook in config['commands'].get('build', '').split("\n"):
  run(hook)

...

for hook in config['commands'].get('deploy', '').split("\n"):
  run(hook)
```

Other things

- Run Drush commands
- Run automated tests
- Verify file permissions
- Restart services
- Anything you can do on the command line…

Fabric has…

- Simplified my build process
- Made my build process more flexible
- Made my build process more robust

- https://www.oliverdavies.uk/talks/deploying-drupal-fabric
- http://fabfile.org
- https://github.com/opdavies/fabric-example-drupal
- https://github.com/opdavies/fabric-example-sculpin
- https://deploy.serversforhackers.com (~~$129~~ $79)

Thanks!

Questions?

@opdavies
oliverdavies.uk