DEPLOYING PHP APPLICATIONS WITH 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 & Sysadmin
- Drupal Bristol, PHPSW, DrupalCamp Bristol
- @opdavies
- oliverdavies.uk

WHAT IS FABRIC?

WHAT IS FABRIC?

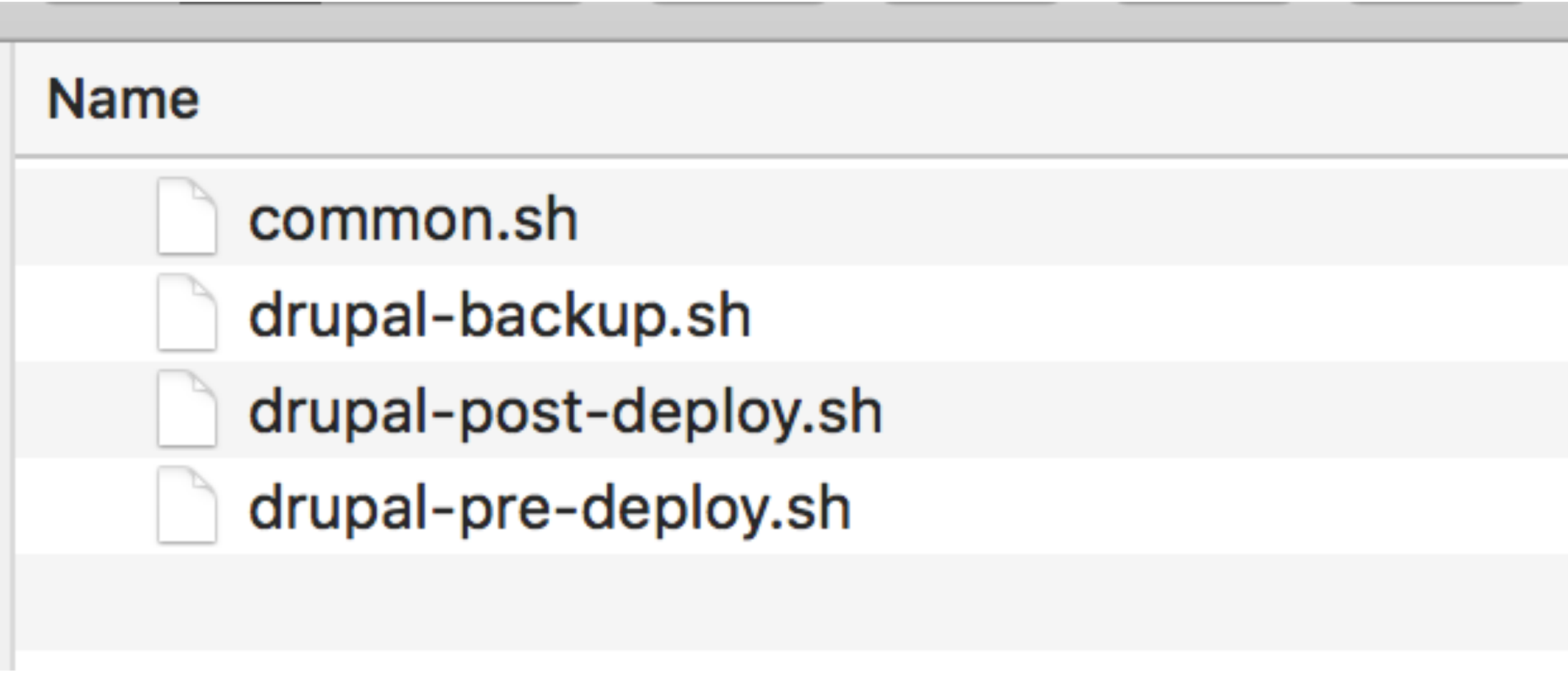
Fabric is a Python (2.5-2.7) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks.

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 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

```
# 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

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

TASK ARGUMENTS

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

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

TASK ARGUMENTS

```
def build(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 build():
    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

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

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

IS THE SITE STILL RUNNING?

CHECKING FOR FAILURES

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

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

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

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

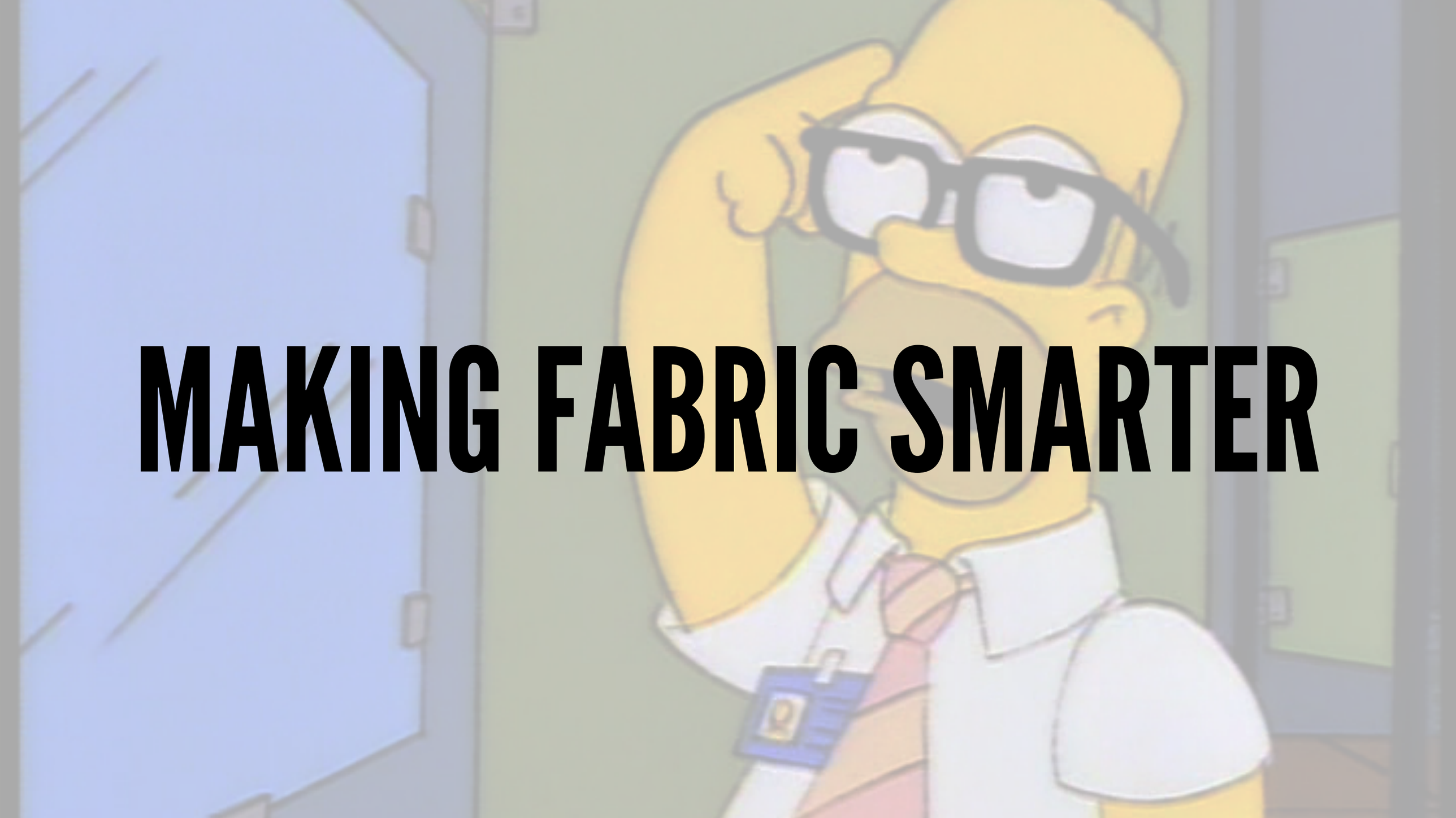
```
$ drush status

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
...
```

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.')
```


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
    theme:
        path: 'themes/custom/drupalbristol'
        build:
            npm: no
            type: gulp
            yarn: yes

composer:
    install: true
```

PROJECT SETTINGS FILE

```
# fabfile.py

from fabric.api import *
import yaml

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']

    with cd(drupal['root']):
        if drupal['version'] == 8:
            if drupal['config']['import'] == True:
                if drupal['config']['cmi_tools']:
                    run('drush cim -y %s' % drupal['config']['import']['name'])
                else:
                    run('drush cimy -y %s' % drupal['config']['import']['name'])

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

PROJECT SETTINGS FILE

```
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/console/artisan commands
- 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-php-fabric
- http://fabfile.org
- https://github.com/opdavies/fabric-example-sculpin
- https://github.com/opdavies/fabric-example-drupal
- https://deploy.serversforhackers.com (~~$129~~ $79)

JOIND.IN/TALK/4E35D

@OPDAVIES

OLIVERDAVIES.UK