

---

# **fusionbox-fabric-helpers**

## **Documentation**

*Release 0.0.1*

**Fusionbox Programmers**

**Jun 12, 2018**



---

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Settings</b>	<b>5</b>
2.1	Getting started . . . . .	5
<b>3</b>	<b>Fabric helpers</b>	<b>7</b>
3.1	Core . . . . .	7
3.2	Django . . . . .	7
3.3	FBORM . . . . .	8
3.4	Utilities . . . . .	8
3.5	Git utilities . . . . .	8
3.6	Update methods . . . . .	8
<b>4</b>	<b>Indices and tables</b>	<b>9</b>
	<b>Python Module Index</b>	<b>11</b>



Contents:



Fabric helpers used by the development team at [Fusionbox](#) for server deployment.

Here is a minimal `fabfile.py`:

```
from fabric.api import env, roles

from fusionbox.fabric import fb_env
from fusionbox.fabric.django import stage, deploy

env.roledefs['live'] = ['cowboyneal@foobar.com']

fb_env.project_name = 'foobar'

stage = roles('dev')(stage)
deploy = roles('live')(deploy)
```

In the case that either the live or dev host has a unique config that is different from the default (such as with webfaction), a fabfile something like this may be used:

```
from fabric.api import env, roles

from fusionbox.fabric import fb_env
from fusionbox.fabric.django import stage, deploy

env.roledefs['live'] = ['cowboyneal@foobar.com']

fb_env.project_name = 'foobar'
fb_env.live_web_home = '/home/cowboyneal/webapps'
fb_env.live_workon_home = '/home/cowboyneal/virtualenvs'

stage = roles('dev')(stage)
deploy = roles('live')(deploy)
```





## 2.1 Getting started

The *fusionbox-fabric-helpers* package exposes a configuration object called *fb\_env*. The *fb\_env* object allows one to customize the behavior of the fabric helpers. For example, the following fabfile uses the *fb\_env* object to set different configurations for the live and dev servers:

```
from fabric.api import env, roles

from fusionbox.fabric import fb_env
from fusionbox.fabric.django import stage, deploy

env.roledefs['live'] = ['foo@bar.com']

fb_env.project_name = 'bar'

fb_env.dev_web_home = '/var/www'
fb_env.dev_workon_home = '/var/python-environments'

fb_env.live_web_home = '/home/foo/webapps'
fb_env.live_workon_home = '/home/foo/virtualenvs'

stage = roles('dev')(stage)
deploy = roles('live')(deploy)
```

This fabfile shows that, on the dev server, the project root directory is located in */var/www* and python virtual environments are located in */var/python-environments*. On the live server, those resources are located in */home/foo/webapps* and */home/foo/virtualenvs* respectively.

Other things are also happening behind the scenes. The *project\_name* setting is used to build the absolute path to the project directory on the dev and live servers as follows:

```
path on dev: /var/www/bar.com
path on live: /home/foo/webapps/bar.com
```

The *project\_name* is also automatically (not automagically!!) used to build the paths to the python virtual environment:

```
path on dev: /var/python-environments/bar
path on live: /home/foo/virtualenvs/bar
```

Any setting which is generated automatically can be manually overridden. If you wanted to manually set the absolute paths to the project, you could do this:

```
fb_env.dev_project_loc = '/var/www/weirdbar.com'
fb_env.live_project_loc = '/home/foo/webapps/unweirdbar.com'
```

## 3.1 Core

## 3.2 Django

`fusionbox.fabric.django.obfuscate()`

Compile all source files to byte code, then remove them.

`fusionbox.fabric.django.obfuscate_decorator(role)`

Given a fabric action, this will run `obfuscate()` after it with config settings for the specified role.

`fusionbox.fabric.django.run_subprocesses(*args, **kws)`

Returns a list of tuples of command, Popen object. During `__close__`, the list of processes is polled for unfinished processes and attempts to close them.

`fusionbox.fabric.django.runserver()`

Runs the local django server, starting up celery workers and/or the solr server if needed.

The following `fb_env` variables must be present in your fabfile for their related processes to be started. Each should be a 2-tuple of directory and command to run.

- `runserver_cmd: ('.', './manage.py runserver')`
- `celery_cmd: ('.', './manage.py celery worker -c 2 --autoreload')`
- `solr_cmd: ('solr', 'java -jar start.jar')`

`fusionbox.fabric.django.shell()`

Fires up a shell on the live server.

`fusionbox.fabric.django.sync_db(role)`

Downloads the latest remote (live or dev) database backup and loads it on your local machine.

`fusionbox.fabric.django.sync_media(role)`

Synchronizes the latest remote (live or dev) media directory with your local media directory.

### 3.3 FBORM

`fusionbox.fabric.fborm.deploy()`  
Same as stage, but always uses the live branch and live config settings.

`fusionbox.fabric.fborm.stage(branch=None, role='dev')`  
Updates the remote site files to your local branch head and migrates.

### 3.4 Utilities

`fusionbox.fabric.utils.files_changed(version, files)`  
Checks if anything in files has changed between version and local HEAD.

`fusionbox.fabric.utils.supervisor_command(action, name)`  
Performs a command on a supervisor process.

`fusionbox.fabric.utils.virtualenv(*args, **kws)`  
Context manager to run all commands under the python virtual env at dir.

### 3.5 Git utilities

`fusionbox.fabric.git.get_git_branch()`  
Returns the name of the active local git branch.

`fusionbox.fabric.git.has_git_branch(branch)`  
Checks if branch is available in the remote git repository.

`fusionbox.fabric.git.is_local_repo_clean()`  
Checks if there are uncommitted changes in the local git repository.

`fusionbox.fabric.git.is_repo_clean()`  
Checks if there are uncommitted changes in the remote git repository.

### 3.6 Update methods

`fusionbox.fabric.update.get_update_function()`  
Returns the update function which will be used to update the remote site files based on the `fb_env.transport_method` config setting.

`fusionbox.fabric.update.update_with_git(branch)`  
Updates the remote git repository to `branch` using git pull.  
  
Returns the commit hash of the remote HEAD before it was updated.

`fusionbox.fabric.update.update_with_rsync(branch)`  
Updates remote site files to local state of `branch` using rsync.  
  
Returns the commit hash of remote version before update.

## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**f**

fusionbox.fabric, 7  
fusionbox.fabric.django, 7  
fusionbox.fabric.fborm, 8  
fusionbox.fabric.git, 8  
fusionbox.fabric.update, 8  
fusionbox.fabric.utils, 8





## D

deploy() (in module fusionbox.fabric.fborm), 8

## F

files\_changed() (in module fusionbox.fabric.utils), 8

fusionbox.fabric (module), 7

fusionbox.fabric.django (module), 7

fusionbox.fabric.fborm (module), 8

fusionbox.fabric.git (module), 8

fusionbox.fabric.update (module), 8

fusionbox.fabric.utils (module), 8

## G

get\_git\_branch() (in module fusionbox.fabric.git), 8

get\_update\_function() (in module fusionbox.fabric.update), 8

## H

has\_git\_branch() (in module fusionbox.fabric.git), 8

## I

is\_local\_repo\_clean() (in module fusionbox.fabric.git), 8

is\_repo\_clean() (in module fusionbox.fabric.git), 8

## O

obfuscate() (in module fusionbox.fabric.django), 7

obfuscate\_decorator() (in module fusionbox.fabric.django), 7

## R

run\_subprocesses() (in module fusionbox.fabric.django), 7

runserver() (in module fusionbox.fabric.django), 7

## S

shell() (in module fusionbox.fabric.django), 7

stage() (in module fusionbox.fabric.fborm), 8

supervisor\_command() (in module fusionbox.fabric.utils), 8

sync\_db() (in module fusionbox.fabric.django), 7

sync\_media() (in module fusionbox.fabric.django), 7

## U

update\_with\_git() (in module fusionbox.fabric.update), 8

update\_with\_rsync() (in module fusionbox.fabric.update), 8

## V

virtualenv() (in module fusionbox.fabric.utils), 8