Cubric Documentation

Release 0.1.0

Matt Wright

Sep 27, 2017

Contents

Cubric allows you to easily create servers, configure servers and deploy arbitrary WSGI applications on Amazon EC2 or Rackspace Cloud Servers using Fabric and Cusine. With a small amount of configuration you can create servers and deploy applications with great ease! For example, you could create a new staging server and deploy your WSGI application to it with the following, single terminal command:

```
$ fab -c rcfile.staging create_server create_app_context deploy
```

Checkout the example application.

CHAPTER 1

Contents

CHAPTER 2

Overview

Cubric has been developed using two core concepts:

Servers

Servers are, you guessed it, servers! Servers are initialized and/or configured to run one or more applications. They can be rebooted, restarted, stopped, etc.

Application Contexts

An application context is a specification for an application. It's sort of the 'house' the app lives in. They are generally specific to a particular type of server and the server's configuration. They can be deployed, stopped, started, restarted, etc.

CHAPTER 3

Getting Started

Install Cubric via pip:

```
$ pip install cubric
```

EC2

Working with EC2 requires boto. Install boto:

```
$ pip install boto
```

Configure your Fabric environment in your fabfile or by using an rcfile by adding the following values:

```
user = ubuntu
key_filename = /path/to/keyfile
#host_string = to be added later

branch = master
cloud_provider = amazon

server_class = cubric.contrib.servers.ubuntu.default.Server
context_class = cubric.contrib.servers.ubuntu.default.WsgiApplicationContext

ec2_key = your_ec2_key
ec2_secret = your_ec2_secret
ec2_ami = ami-fd589594
ec2_keypair = keypair_name
ec2_secgroups = security_group_name
ec2_instancetype = t1.micro
```

Rackspace

Install python-cloudservers:

```
$ pip install python-cloudservers
```

Configure your Fabric environment in your fabfile or by using an rcfile by adding the following values:

```
user = root
#password = to be added later
#host_string = to be added later

branch = master
cloud_provider = amazon

server_class = cubric.contrib.servers.ubuntu.default.Server
context_class = cubric.contrib.servers.ubuntu.default.WsgiApplicationContext

rackspace_username = your_username
rackspace_apikey = your_api_key
rackspace_servername = your_server_name
rackspace_image = 112
rackspace_flavor = 1
```

Create a Server

Create a file named *fabfile.py* in the root of your project and add the following:

```
from cubric.tasks import *
```

To see a list of tasks you can run the command:

```
$ fab --list
```

Now go ahead and create a server:

```
$ fab -c your_rcfile create_server
```

This will create and initialize your server. Be sure to make note of the public DNS or IP address. If you've already created your server through your respective management console, or if something fails during server setup, you can initialize the server by setting the value for *host_string* and either *key_filename* or *password* (depending on you access your server over SSH) in your rcfile. Then run the following command:

```
$ fab -c your_rcfile initialize_server
```

Create an Application Context

An application context is the environment under which an application runs. Cubric is primarily developed for Python applications. More specifically WSGI applications that are run using nginx, uWSGI, and supervisor. To create and application context on the server run the following command:

```
$ fab -c your_rcfile create_app_context
```

Now you should be ready to deploy your WSGI application to the server if your application is setup to play nice with the specified application context.

Deploy Your App

As previously mentioned, the default server and application context that ships with Cubric uses nginx, uWSGI, and supervisor. The application context expects the following things:

1. The application is under version control using git
2. A fabric environment variable named *branch* to be set to the branch you wish to deploy such as *develop* or *origin/master*
3. A fabric environment variable named *context__environment* to be set to your desired environment name such as *development*, *staging*, etc. It defaults to *development* if you don't set it.
3. Two configuration templates be present in your application where <*environment_name*> is equal to the *context__environment* Fabric environment variable:

   (a) *etc/<environment_name>/nginx.conf.tmpl*

   (b) *etc/<environment_name>/supervisor.conf.tmpl*
4. A WSGI entry file at the root of your project named *wsgi.py* that includes the WSGI callable named *app*. If you want to override the defaults you can set two Fabric environment variables named *context__wsgi_file* and *wsgi_callable*.

You can refer to the example project to see these requirements in action.

Then, to deploy your application, simply run the following command:

```
$ fab -c your_rcfile deploy
```

The Almighty One Liner

If you're application is configured properly for the application context you could quickly create a server and deploy your app using the following command:

```
$ fab -c your_rcfile create_server create_app_context deploy
```

Hooray!

CHAPTER 4

Changelog

Cubric Changelog

Here you can see the full list of changes between each Cubric release.

Version 1.1.0

Initial release. July 18th 2012

CHAPTER 5

Indices and tables

- genindex
- modindex
- search