Flying Circus

Release 0.7.2-beta

Gary Donovan

Oct 29, 2019

CONTENTS:

Flying Circus is a tool for DevOps engineers who use Amazon Web Services. It extends AWS CloudFormation with a Pythonic interface, to provide a more convenient and powerful way to describe AWS infrastructure as code.

CHAPTER ONE

INTRODUCTION

Putting more *Code* into your *Infrastructure as Code*

Flying Circus is a tool for describing AWS infrastructure as code (using Python). It uses the same data structures as the AWS Cloud Formation service, except described as Python objects instead of the usual YAML. The Python program generates a YAML template, which is passed across to Cloud Formation in the usual manner.

It is a bit unusual to use a full programming language to describe infrastructure, instead of a static configuration file like many of us are used to (whether or not we also utilise a templating tool). We hope that the Flying Circus library can empower DevOps folk by unlocking some of the techniques that are available for software code, like named variables and techniques to structure code independently of the output format, libraries to allow code re-use with versioning, automated refactoring tools and so on.

CHAPTER

TWO

GETTING STARTED

2.1 Installation

Install Flying Circus through the Python packaging system:

```
pip install flying-circus
```

Many people also use the Amazon Web Services command line tools to deploy their CloudFormation stacks. The Python packaging system is a good way to install an up-to-date version of these too:

```
# Optional
pip install awscli
```

2.2 Example

Here is a simple example of how you can use Flying Circus to describe some EC2 instances and deploy them using the AWS CloudFormation service.

First, create a python script (called *ec2_example.py* in this case) to describe your infrastructure. Any valid Python can be used to create the Flying Circus objects, along with any valid CloudFormation properties and attributes.

This example is intentionally simplistic - it just creates two EC2 instances with varying configuration, and outputs the internal IP address for one. However, it does hint at some of the more complex and powerful usage patterns.

```
#!/usr/bin/env python3

import os

from flyingcircus.core import Output
from flyingcircus.intrinsic_function import GetAtt
from flyingcircus.service.ec2 import *


def create_ec2_instance(name, instance_type="t2.micro"):
    instance = Instance(Properties=InstanceProperties(
        ImageId="ami-942dd1f6",
        InstanceType=instance_type,
        Monitoring=False,
    ))
    instance.name = name
    return instance
```

(continues on next page)

(continued from previous page)

```python
def generate_stack_template():
    stack = Stack()

    stack.Resources["WebServer"] = create_ec2_instance("webserver")

    stack.Resources["DatabaseServer"] = dbserver = create_ec2_instance("dbserver",
↪"t2.medium")
    dbserver.DeletionPolicy = "Retain"

    stack.Outputs["DatabaseServerIp"] = Output(
        Description=f"Internal IP address for the database server",
        Value=GetAtt(dbserver, "PrivateIp"),
    )

    stack.tag(application="api-service", environment="test", owner=os.environ.get(
↪"USER"))

    return stack.export("yaml")


if __name__ == "__main__":
    print(generate_stack_template())
```

Now generate CloudFormation YAML from your Python script. Note that the result will *always* be valid well-formatted YAML, and internal checks mean that it is also difficult to generate invalid CloudFormation.

Finally, use the AWS command line tools to create/update a stack and it's associated resources (assuming you have configured your AWS credentials. . .).

```
python ec2_example.py > ec2_example.yaml
aws cloudformation deploy --stack-name flying-circus-ec2-example --template-file ec2_
↪example.yaml
```

These last steps are an obvious candidate to go in your Continuous Integration server ;-)

CHAPTER

THREE

LICENCING - LGPL

3.1 What?

Flying Circus and its code is made available under the LGPL (GNU Lesser General Public Licence), version 3. This is a copy-left open source licence intended for libraries that are used by other projects, regardless of the licencing that project uses.

The expectation is that if you modify Flying Circus itself (eg. fix a bug in the core components, or update the mappings to Amazon Web Services) then you will make those improvements available to the broader community, ideally via a Pull Request on GitHub. When you use Flying Circus to define your AWS infrastructure as a separate project or Python module, then that Python code and any generated artefacts (like Cloud Formation YAML) remains your own, which you can do with as you please.

3.2 Why?

The open source community nowadays is a fantastic example of something that is greater than the sum of it's parts. So much modern software is built on the shoulders of giants - our predecessors and peers who have created powerful software for us to use. But there will always be selfish people who take advantage of everyone else, and don't want to give back to the community.

A GPL-style licence provides a degree of legal protection to force bad actors to do the right thing, whilst not inconveniencing the rest of us. The LGPL variant, in particular, allows users to retain IP ownership and confidentiality for their internal infrastructure, whilst still gaining the power of Flying Circus.

Thanks for reading. We sincerely hope that Flying Circus helps you, and look forward to hearing from you in our community.

3.3 Third Party

Flying Circus functionality is implemented using 3rd party libraries that have their own licences. It is your responsibility to check that these licences are acceptable to you.

Flying Circus is used in conjunction with Amazon Web Services, as a tool for using the AWS CloudFormation API. As such, Flying Circus is partially dependent on the AWS CloudFormation Template Resource specification . The current version of the specification referenced by Flying Circus can be found in the `contrib/` directory in the project's source code.