awslambdahelper Documentation

Release 1.1.10

Drew J. Sonne

Jun 25, 2017

Contents

1 Installation **3**

2 QuickStart **5**

2.1 Create a Python class 5

2.2 Setup AWS Lambda handler 5

2.3 Create AWS Config Rule 5

Abstracts the more mundane aspects of lambda resources

A lot of boilerplate code is required to implemented lambda's for AWS Config and custom Cloudformation resources. We can abstract this away and wrap our rule in data structures to improve development and encourage a particular structure.

CHAPTER 1

Installation

```
$ pip install awslambdahelper
```

CHAPTER 2

QuickStart

Create a Python class

```
# my_lambda_code.py
from awslambdahelper import AWSConfigRule

class MyConfigRule(AWSConfigRule):
    def find_violation_config_change(self, config, rule_parameters):
      return [NonCompliantEvaluation(
        Annotation="This failed because it is only a demo."
      )]
```

Setup AWS Lambda handler

```
>>> import boto3
>>> boto3.client('lambda').create_function(
  Handler = "my_lambda_code.MyConfigRule.handler'
)
```

Create AWS Config Rule

Getting Started with Custom Rules.

That's it! For a more indepth guide, read the docs.

Developer Guide

AWS Config Rule

AWS Config rules come in two flavours: _Scheduled_ and _ConfigurationChange_.

Scheduled rules are invoked by AWS Config on a periodic basis as defined in your rule, and _ConfigurationChange_ rules are invoked by AWS Config when a configuration change occurs. Generally, you used scheduled rules for resources which AWS Config [support directly](http://docs.aws.amazon.com/config/latest/developerguide/resource-config-reference.html), and _ConfigurationChange_ rules for [additional resources types for AWS Config](http://docs.aws.amazon.com/config/latest/developerguide/evaluate-config_develop-rules_nodejs.html#creating-custom-rules-for-additional-resource-types).

Create a new class, write it in a function to be set as the lambda handler, and override either the *find_violation_scheduled(...)* function or *find_violation_config_change(...)*.

Configuration Change Rule

‘‘‘python from awsconfig_lambdahelper.configrule import AWSConfigRule from awsconfig_lambdahelper.evaluation import CompliantEvaluation,NonCompliantEvaluation

A schedule AWS config rule class MyCustomConfigurationChangeRule(AWSConfigRule):

def find_violation_config_change(self, config, rule_parameters):

rule_responses = apply_my_rule_to_a_resource(config)

response = [] for violation in rule_responses:

if violation[’failed’]:

response.append(

NonCompliantEvaluation(Annotation=”This failed because of a good reason.”

)

)

else: # There’s no need to set the resource id or type, as the library is aware of those # values and will apply them automatically. response.append(CompliantEvaluation())

return response

Lambda entrypoint def lambda_handler(event, context):

my_rule = MyCustomConfigurationChangeRule(applicable_resources=[”AWS::EC2::Instance”]

) my_rule.lambda_handler(event, context)

‘‘‘

Scheduled Rule

‘‘‘python

from awsconfig_lambdahelper.configrule import AWSConfigRule from awsconfig_lambdahelper.evaluation import CompliantEvaluation,NonCompliantEvaluation

A schedule AWS config rule class MyCustomScheduledConfigRule(AWSConfigRule):

def find_violation_scheduled(self, ruleParameters, accountId):

rule_responses = apply_my_rules()

response = [] for violation in rule_responses:

```
            if violation['failed']:
                response.append(  # Scheduled rules are not in response to a config change, so you need to tell AWS Config what # resources you were looking at. CompliantEvaluation(
                    ResourceType=violation['my_resource_type'], ResourceId=violation['my_resource_id']
                )
                )
            else:
                response.append(
                NonCompliantEvaluation(  ResourceType=violation['my_resource_type'], ResourceId=violation['my_resource_id'], Annotation="This failed because of a good reason."
                )
                )
        return response
# Lambda entrypoint def lambda_handler(event, context):
    my_rule = MyCustomScheduledConfigRule(  applicable_resources=["AWS::EC2::Instance"]
    ) my_rule.lambda_handler(event, context)
```

‘‘‘

API Reference

Rules

Evaluations

CompliantEvaluation

NonCompliantEvaluation

NotApplicableEvaluation

InsufficientDataEvaluation

Private Class and Function Reference

lambdahelper-bunder cli tool

AWS Config resources

What’s new in awslambdahelper 1.1.10

Release: 1.1.10

Date: 25 June 2017

Changes since 1.1.6

- Expanded test coverage
- Expanded the documentation
- Refactored the cli tool to improve test coverage

Release: 1.1.6

Date: 24 June 2017

Changes since 1.0.0

- Expanded test coverage
- Expanded the documentation and added a short user guide.
- Restructured the module to export `AWSConfigRule`, `CompliantEvaluation`, `NonCompliantEvaluation`, `NotApplicableEvaluation`, `InsufficientDataEvaluation`