webhook2lambda2sqs Documentation

Release 0.2.0

Jason Antman

Jun 26, 2017

Contents

Generate code and manage infrastructure for receiving webhooks with AWS API Gateway and pushing to SQS via Lambda.

Webhooks are great, and many projects and services are now offering them as a notification option. But sometimes it makes more sense to have the messages in a queue that can absorb changes in rate and de-couple the sending service from a potentially slow or unavailable backend.

webhook2lambda2sqs generates code for an AWS Lambda function to receive webhook content via API Gateway and push it to an SQS queue, where it can be consumed as needed. It supports multiple endpoints via unique URL paths (API Gateway resources), where content sent to each endpoint is pushed to one or more SQS queues.

In addition, webhook2lambda2sqs includes a wrapper around HashiCorp Terraform to automate creation and maintenance of all or part of the infrastructure required to operate this (the API Gateway and its configuration, the Lambda function, the function's IAM role, etc.). If TerraForm isn't a viable option for you to manage infrastructure with, you can use the generated configuration (which maps quite closely to AWS API parameters) as a guide for manual management.

There are also helper commands to view the Lambda Function and API Gateway logs, send a test message, and view the queue contents.

CHAPTER 1

Program Components

- Lambda Function code generation
- Terraform configuration generation
- Management of infrastructure via Terraform
- AWS-related helpers for inspecting queues and logs, querying information, and enabling metrics/logging/rate limiting on the API Gateway.

CHAPTER 2

Bugs and Feature Requests

Bug reports and feature requests are happily accepted via the GitHub Issue Tracker. Pull requests are welcome. Issues that don't have an accompanying pull request will be worked on as my time and priority allows.

CHAPTER 3

Contents

Requirements and Installation

Requirements

Requirements

- An Amazon AWS account to run this all in (note - it will probably be cheap, but not free)
- Python 2.7+ (currently tested with 2.7, 3.3, 3.4, 3.5). Note that AWS Lambda currently only supports python 2.7 as an execution environment, but you're welcome to use other versions on the machine where you run this project.
- Python VirtualEnv and `pip` (recommended installation method; your OS/distribution should have packages for these)
- HashiCorp Terraform >= 0.6.16 to manage the AWS infrastructure, if desired. Terraform is written in Go, and distributed as a static binary.

Installation

It's recommended that you install into a virtual environment (virtualenv / venv). See the virtualenv usage documentation for information on how to create a venv. If you really want to install system-wide, you can (using sudo).

```
pip install webhook2lambda2sqs
```

If you wish to use Terraform to manage the infrastructure, you need to install that according to the documentation. Note that there are packages available in the official repositories of most Linux distributions, and there is also a Homebrew formula for Mac users.

Configuration

webhook2lambda2sqs is configured via a JSON configuration file, which defines both settings for Terraform to manage the infrastructure, as well as the mapping of API Gateway URL paths to SQS queues. You can view a sample configuration file as well as documentation on the various fields with `webhook2lambda2sqs example-config`; the config file example will be written to STDOUT (so it may be redirected to a file) and the documentation will be written to STDERR.

Example output of the `example-config` action:

```
$ webhook2lambda2sqs example-config
{
    "api_gateway_method_settings": {
        "dataTraceEnabled": false,
        "loggingLevel": "OFF",
        "metricsEnabled": false,
        "throttlingBurstLimit": null,
        "throttlingRateLimit": null
    },
    "deployment_stage_name": "something",
    "endpoints": {
        "other_resource_path": {
            "method": "GET",
            "queues": [
                "queueName2",
                "queueName3"
            ]
        },
        "some_resource_path": {
            "method": "POST",
            "queues": [
                "queueName1",
                "queueName2"
            ]
        }
    },
    "logging_level": "INFO",
    "name_suffix": "something",
    "terraform_remote_state": {
        "backend": "backend_name",
        "config": {
            "option_name": "option_value"
        }
    }
}

Configuration description:

api_gateway_method_settings - (optional) Dictionary of API Gateway Method
  settings to enable. See
  <https://docs.aws.amazon.com/apigateway/api-reference/resource/stage/
↪#methodSettings>
  for upstream documentation. Due to a limitation
  in Terraform (https://github.com/hashicorp/terraform/issues/6612), these
  settings are applied by this program via the AWS API after Terraform has
  run; if you use this program to generate Terraform configurations and
  apply them yourself, these settings will have no effect. The following
  keys and values are supported:
```

```
  - 'metricsEnabled' - (boolean, default False) whether or not to enable
    CloudWatch metrics for the API.
  - 'loggingLevel' - (string, default "OFF") logging level to use for
    pushing API Gateway logs to CloudWatch Logs. Valid values are "OFF",
    "ERROR" or "INFO".
  - 'dataTraceEnabled' - (boolean, default False) whether to enable data
    trace logging to CloudWatch Logs for the API Gateway.
  - 'throttlingBurstLimit' - (integer, default None) API Gateway throttling
    burst limit - see:
    <http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-
↪throttling.html?icmpid=docs_apigateway_console>
    Omit to not set this option.
  - 'throttlingRateLimit' - (double, default None) API Gateway throttling
    rate limit (requests per second). See:
    http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-
↪throttling.html?icmpid=docs_apigateway_console
    Omit to not set this option.

deployment_stage_name - (optional) String used as the name for the API
  Gateway Deployment Stage, which will be the beginning component of the
  URL path for the API Gateway
  (i.e. https://<api id>.execute-api.us-east-1.amazonaws.com/STAGE_NAME/).
  Defaults to "webhook2lambda2sqs".

endpoints - dict describing each webhook endpoint to setup in API Gateway.
  - key is the API Gateway resource name (final component of the URL)
  - value is a dict with the following keys:
    - 'method' - HTTP method for API Gateway resource
    - 'queues' - list of SQS queue names to push request content to

logging_level - the Python logging level (constant name) to set for the
  lambda function. Defaults to INFO. Currently the function only logs at
  ERROR and DEBUG levels.

name_suffix - (optional) by default, all AWS resources will be named
  "webhook2lambda2sqs"; specify a suffix to add to that name here.

terraform_remote_state - (optional) dict of Terraform remote state options.
  If specified, will call 'terraform remote config' before every terraform
  command to setup remote state storage. See:
  https://www.terraform.io/docs/state/remote/index.html

  Dict keys:
  - 'backend' - name of the terraform remote state backend to configure
  - 'config' - dict of backend configuration option name/value pairs
```

Note About API Gateway Method Settings

Terraform currently lacks support for setting the `methodSettings` on an API Gateway Stage. These settings are used to control rate limiting as well as enable CloudWatch logs and metrics for the API Gateway itself. Until Terraform adds support for this, if you specify the `api_gateway_method_settings` configuration key, this program will apply the relevant settings using the AWS API directly after the Terraform run is complete.

Usage

Generating Code and Infrastructure Configuration

To only generate the Lambda function code and Terraform configuration, run:

```
$ webhook2lambda2sqs generate
```

Note that when applying the configuration outside of `webhook2lambda2sqs` (i.e. using terraform directly), the `api_gateway_method_settings` configuration key will be ignored. See *Note About API Gateway Method Settings*.

Note that the generated Terraform is a single file and does not make use of variables. As Terraform doesn't support iteration or conditionals, it's really required that we generate the important parts of the configuration programmatically, so there's little use in `tfvars`.

Managing Infrastructure

`webhook2lambda2sqs` provides `plan`, `apply` and `destroy` commands which wrap Terraform runs to perform those actions on the generated configuration. There is also a `genapply` command to generate the Lambda Function and Terraform configuration, and then apply it all in one command.

You'll want to have the `AWS_DEFAULT_REGION` environment variable set. AWS credentials are managed however you want per terraform's documentation, i.e. environment variables, shared credentials file or using an instance profile/role on an EC2 instance.

Important Note: Unlike CloudFormation, Terraform relies on storing the state of your managed infrastructure itself. You can use a variety of remote backends storage options including Consul, etcd, http and S3, or you can leave the default of storing state locally in a `terraform.tfstate` file. Please note that you'll need to save state somewhere in order to update or destroy the infrastructure you created. You can specify remote state options in the configuration file, or just deal with the state file locally.

AWS Resources Created

The following AWS resources are created by this program, when using a relatively default configuration:

- 2 IAM Roles (the role used by API Gateway to invoke the Lambda Function, and the role used by the Lambda Function itself)
- 2 IAM Role Policy Attachments, one for each of those roles
- 1 API Gateway ReST API
- 1 Lambda Function
- 2 API Gateway Integration Responses for each method type (GET or POST) used
- 1 API Gateway Deployment
- 2 API Gateway Models

And for each endpoint in the configuration file:

- 2 API Gateway Method Responses for each configured endpoint
- An API Gateway Resource for each configured endpoint
- An API Gateway Integration for each configured endpoint

- An API Gateway Method for each configured endpoint

Estimated Cost of Infrastructure

The following provides a cost estimate of the infrastructure as of August 6, 2016 in the US-East region. Please use the links below to find updated pricing information.

- Lambda Function
  - First 1 million Lambda requests per month are free; $0.20 per million after that.
  - Compute time is charged at $0.000000208 per 100ms, with the first 3,200,000 seconds free (this function runs with the default minimum of 128MB memory). Casual testing shows that this function, when pushing to one queue, should execute in 100-1100ms.
- API Gateway
  - $3.50 per million API calls (first 1 million per month are free for the first 12 months)
  - $0.09/GB for the first 10 TB of data transfer

Required IAM Permissions For Code Generation

Generating the Terraform configuration files requires the `iam::GetUser` permission for the user you're running it as. This is required to determine your AWS account ID, which is needed in the IAM policy. In addition, the region that you connect with will be included in the policy.

Required IAM Permissions For Infrastructure Management and Querying

Managing the infrastructure via Terraform, and using the AWS helper actions (such as `queuepeek`, `logs` and `apilogs`) requires the following IAM permissions:

```
apigateway:CreateDeployment
apigateway:CreateModel
apigateway:CreateResource
apigateway:CreateRestApi
apigateway:DeleteIntegration
apigateway:DeleteIntegrationResponse
apigateway:DeleteMethod
apigateway:DeleteMethodResponse
apigateway:DeleteModel
apigateway:DeleteResource
apigateway:DeleteRestApi
apigateway:DeleteStage
apigateway:GetDeployment
apigateway:GetIntegration
apigateway:GetIntegrationResponse
apigateway:GetMethod
apigateway:GetMethodResponse
apigateway:GetModel
apigateway:GetResource
apigateway:GetResources
apigateway:GetRestApi
apigateway:GetRestApis
apigateway:GetStage
apigateway:PutIntegration
apigateway:PutIntegrationResponse
```

```
apigateway:PutMethod
apigateway:PutMethodResponse
apigateway:UpdateStage
iam:CreateRole
iam:DeleteRole
iam:DeleteRolePolicy
iam:GetRole
iam:GetRolePolicy
iam:GetUser
iam:ListInstanceProfilesForRole
iam:PutRolePolicy
lambda:CreateFunction
lambda:DeleteFunction
lambda:GetFunction
lambda:UpdateFunctionCode
lambda:UpdateFunctionConfiguration
logs:DescribeLogGroups
logs:DescribeLogStreams
logs:GetLogEvents
sqs:DeleteMessage
sqs:GetQueueAttributes
sqs:GetQueueUrl
sqs:ListQueues
sqs:ReceiveMessage
```

HTTP Responses

The API Gateway methods respond with a JSON object including `status`, `request_id` (API Gateway Request ID) and `message` fields. If any messages were successfully enqueued, the SQS MessageIds will be available in a `SQSMessageIds` list. Successful, or partially successful requests get a 202 response code, and complete failures get a 500 response code.

Completely successful response

```
{
  "status" : "success",
  "message" : "enqueued 1 messages",
  "SQSMessageIds": ["0720e7b5-8a81-4258-ba6c-afd69bcf60f6"],
  "request_id": "37af7edd-5bf2-11e6-9dcf-19b7d04d8b74"
}
```

Partially successful response

```
{
  "status" : "partial",
  "message" : "enqueued 1 messages; 1 failed",
  "SQSMessageIds": ["549eda2f-b449-4e2a-908c-ab9bb4a8022d"],
  "request_id": "b11a1b6b-5bf2-11e6-8fdb-a3f21465c2f6"
}
```

Failure response

```
{
  "status" : "error",
  "message" : "Exception: Failed enqueueing all messages",
  "request_id": "505e01a7-5bf2-11e6-91eb-adc915445063"
}
```

Queue Message Format

The lambda function will enqueue JSON messages in the SQS queue(s) containing all information available to the function; namely, the `event` dictionary describing the event that triggered the function and the context object (minus any un-serializable attributes) describing the execution environment.

Additionally, a top-level `data` key will be created in the message, containing the POST data for POST requests or the parameters for GET requests.

GET

Example GET request against endpoint name `other` (configured to enqueue in one queue) of ReST API ID `hc0zenxxs1`, with configuration parameter `deployment_stage_name` set to `hooks`:

```
$ curl -i 'https://hc0zenxxs1.execute-api.us-east-1.amazonaws.com/hooks/other/?
↪foo=bar&baz=blam'
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 190
Connection: keep-alive
Date: Sat, 06 Aug 2016 16:59:22 GMT
x-amzn-RequestId: 1df19527-5bf7-11e6-9962-f9b69e643a7f
X-Cache: Miss from cloudfront
Via: 1.1 33ca1aa308d2c3dd926101c0bb80980a.cloudfront.net (CloudFront)
X-Amz-Cf-Id: pXs73XR10oo6yzAjcopq0c6ZNYGNF3NntEVKAyVFp2HdcEsRikCqtQ==

{
  "status" : "success",
  "message" : "enqueued 1 messages",
  "SQSMessageIds": ["74219fc5-99cb-4f07-8e86-df4793ebe2ee"],
  "request_id": "1df19527-5bf7-11e6-9962-f9b69e643a7f"
}
```

Message enqueued for that request (MessageId `74219fc5-99cb-4f07-8e86-df4793ebe2ee`):

```
{
    "context": {
        "aws_request_id": "1dfd7bb5-5bf7-11e6-960b-f76084d997b6",
        "client_context": null,
        "function_name": "webhook2lambda2sqs",
        "function_version": "$LATEST",
        "invoked_function_arn": "arn:aws:lambda:us-east-1:<AWS_ACCOUNT_ID>
↪:function:webhook2lambda2sqs",
        "log_group_name": "/aws/lambda/webhook2lambda2sqs",
        "log_stream_name": "2016/08/06/[$LATEST]1c9be741f09e489b852d6f0932e5d15c",
        "memory_limit_in_mb": "128"
```

```
    },
    "data": {
        "baz": "blam",
        "foo": "bar"
    },
    "event": {
        "body-json": {},
        "context": {
            "account-id": "",
            "api-id": "hc0zenxxs1",
            "api-key": "",
            "authorizer-principal-id": "",
            "caller": "",
            "cognito-authentication-provider": "",
            "cognito-authentication-type": "",
            "cognito-identity-id": "",
            "cognito-identity-pool-id": "",
            "http-method": "GET",
            "request-id": "1df19527-5bf7-11e6-9962-f9b69e643a7f",
            "resource-id": "dpc2fo",
            "resource-path": "/other",
            "source-ip": "24.98.234.117",
            "stage": "hooks",
            "user": "",
            "user-agent": "curl/7.49.1",
            "user-arn": ""
        },
        "params": {
            "header": {
                "Accept": "*/*",
                "CloudFront-Forwarded-Proto": "https",
                "CloudFront-Is-Desktop-Viewer": "true",
                "CloudFront-Is-Mobile-Viewer": "false",
                "CloudFront-Is-SmartTV-Viewer": "false",
                "CloudFront-Is-Tablet-Viewer": "false",
                "CloudFront-Viewer-Country": "US",
                "Host": "hc0zenxxs1.execute-api.us-east-1.amazonaws.com",
                "User-Agent": "curl/7.49.1",
                "Via": "1.1 33ca1aa308d2c3dd926101c0bb80980a.cloudfront.net␣
↪(CloudFront)",
                "X-Amz-Cf-Id": "9_xjz8J2zxic_S9QUudB1k8oiw_0IoIlgLXqVKyzapSbg-
↪AXcxEjIg==",
                "X-Forwarded-For": "24.98.234.117, 216.137.42.113",
                "X-Forwarded-Port": "443",
                "X-Forwarded-Proto": "https"
            },
            "path": {},
            "querystring": {
                "baz": "blam",
                "foo": "bar"
            }
        },
        "stage-variables": {}
    }
}
```

POST

Example POST request against endpoint name `some_resource_name` (configured to enqueue in two queues) of ReST API ID `hc0zenxxs1`, with configuration parameter `deployment_stage_name` set to `hooks`:

```
$ curl -i -X POST -H 'Content-Type: application/json' -d '{"foo": "bar", "baz": "blam
↪"}' 'https://hc0zenxxs1.execute-api.us-east-1.amazonaws.com/hooks/some_resource_
↪name/'
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 229
Connection: keep-alive
Date: Sat, 06 Aug 2016 17:07:11 GMT
x-amzn-RequestId: 36a79c5a-5bf8-11e6-928c-cf1c3c738a1d
X-Cache: Miss from cloudfront
Via: 1.1 5b1f6dfc9ebdbec2869a5bfa561dded0.cloudfront.net (CloudFront)
X-Amz-Cf-Id: _Gpuo8bQJrn0Iniz5IsP4BurnCqUDEtLcsXFx1buoneNOYhxYprOUg==


{
  "status" : "success",
  "message" : "enqueued 2 messages",
  "SQSMessageIds": ["6bf3600d-2734-4b31-bed3-c996a0290e09","c29320f7-3687-4652-b0f4-
↪ffea02052ea2"],
  "request_id": "36a79c5a-5bf8-11e6-928c-cf1c3c738a1d"
}
```

Message enqueued for that request (MessageId `6bf3600d-2734-4b31-bed3-c996a0290e09`):


```
{
    "context": {
        "aws_request_id": "36b30df4-5bf8-11e6-a360-8be4c8dc98ef",
        "client_context": null,
        "function_name": "webhook2lambda2sqs",
        "function_version": "$LATEST",
        "invoked_function_arn": "arn:aws:lambda:us-east-1:<AWS_ACCOUNT_ID>
↪:function:webhook2lambda2sqs",
        "log_group_name": "/aws/lambda/webhook2lambda2sqs",
        "log_stream_name": "2016/08/06/[$LATEST]1c9be741f09e489b852d6f0932e5d15c",
        "memory_limit_in_mb": "128"
    },
    "data": {
        "baz": "blam",
        "foo": "bar"
    },
    "event": {
        "body-json": {
            "baz": "blam",
            "foo": "bar"
        },
        "context": {
            "account-id": "",
            "api-id": "hc0zenxxs1",
            "api-key": "",
            "authorizer-principal-id": "",
            "caller": "",
            "cognito-authentication-provider": "",
            "cognito-authentication-type": "",
```

```
            "cognito-identity-id": "",
            "cognito-identity-pool-id": "",
            "http-method": "POST",
            "request-id": "36a79c5a-5bf8-11e6-928c-cf1c3c738a1d",
            "resource-id": "mgaaye",
            "resource-path": "/some_resource_name",
            "source-ip": "24.98.234.117",
            "stage": "hooks",
            "user": "",
            "user-agent": "curl/7.49.1",
            "user-arn": ""
        },
        "params": {
            "header": {
                "Accept": "*/*",
                "CloudFront-Forwarded-Proto": "https",
                "CloudFront-Is-Desktop-Viewer": "true",
                "CloudFront-Is-Mobile-Viewer": "false",
                "CloudFront-Is-SmartTV-Viewer": "false",
                "CloudFront-Is-Tablet-Viewer": "false",
                "CloudFront-Viewer-Country": "US",
                "Content-Type": "application/json",
                "Host": "hc0zenxxs1.execute-api.us-east-1.amazonaws.com",
                "User-Agent": "curl/7.49.1",
                "Via": "1.1 5b1f6dfc9ebdbec2869a5bfa561dded0.cloudfront.net␣
↪(CloudFront)",
                "X-Amz-Cf-Id": "cFDJLUHTujRZpIKnCtFhklW5XI3vMU7mz7ADpg52J5R5Mqklo4-
↪hdg==",
                "X-Forwarded-For": "24.98.234.117, 216.137.42.98",
                "X-Forwarded-Port": "443",
                "X-Forwarded-Proto": "https"
            },
            "path": {},
            "querystring": {}
        },
        "stage-variables": {}
    }
}
```


webhook2lambda2sqs

webhook2lambda2sqs package

Submodules

webhook2lambda2sqs.aws module

webhook2lambda2sqs.config module

class webhook2lambda2sqs.config.**Config**(*path*)

Bases: object

_allowed_methods = ['POST', 'GET']

_example = {'terraform_remote_state': {'config': {'option_name': 'option_value'}, 'backend': 'backend_name'}, 'api_g

_example_docs = '\n Configuration description:\n\n api_gateway_method_settings - (optional) Dictionary of API Gatev

_load_config(*path*)
Load configuration from JSON

Parameters **path** (*str*) – path to the JSON config file

Returns config dictionary

Return type dict

_validate_config()
Validate configuration file. :raises: RuntimeError

static **example_config**()
Return a 2-tuple of example configuration file as a pretty-printed JSON string and documentation about it as a string.

Return type tuple

Returns 2-tuple of (example config file as pretty-printed JSON string, documentation about it (str))

func_name
Return what we will name our lambda function.

Returns lambda function name

Return type str

get(*key*)
Get the value of the specified configuration key. Return None if the key does not exist in the configuration.

Parameters **key** (*str*) – name of config key

Returns configuration value at specified key

Return type object

logging_level
Return the string name of the logging module level constant to set in the lambda function.

Returns logging level constant name

Return type str

stage_name
Return the string to use for our API Gateway Deployment Stage name.

Returns API Gateway Deployment Stage name

Return type str

exception webhook2lambda2sqs.config.**InvalidConfigError**(*message*)
Bases: exceptions.Exception

Raised for configuration errors

webhook2lambda2sqs.func_generator module

webhook2lambda2sqs.json_templates module

webhook2lambda2sqs.lambda_func module

webhook2lambda2sqs.runner module

webhook2lambda2sqs.terraform_runner module

class webhook2lambda2sqs.terraform_runner.**TerraformRunner**(*config*, *tf_path*)

Bases: object

_args_for_remote()

Generate arguments for 'terraform remote config'. Return None if not present in configuration.

Returns list of args for 'terraform remote config' or None

Return type list

_get_outputs()

Return a dict of the terraform outputs.

Returns dict of terraform outputs

Return type dict

_run_tf(*cmd*, *cmd_args=[]*, *stream=False*)

Run a single terraform command via *run_cmd()*; raise exception on non-zero exit status.

Parameters

- **cmd** (*str*) – terraform command to run
- **cmd_args** (list) – arguments to command

Returns command output

Return type str

Raises Exception on non-zero exit

_set_remote(*stream=False*)

Call *_args_for_remote()*; if the return value is not None, execute 'terraform remote config' with those arguments and ensure it exits 0.

Parameters **stream** (*bool*) – whether or not to stream TF output in realtime

_setup_tf(*stream=False*)

Setup terraform; either 'remote config' or 'init' depending on version.

_show_outputs()

Print the terraform outputs.

_taint_deployment(*stream=False*)

Run 'terraform taint aws_api_gateway_deployment.depl' to taint the deployment resource. This is a workaround for https://github.com/hashicorp/terraform/issues/6613

Parameters **stream** (*bool*) – whether or not to stream TF output in realtime

_validate()

Confirm that we can run terraform (by calling its version action) and then validate the configuration.

apply(*stream=False*)

Run a 'terraform apply'

Parameters **stream** (*bool*) – whether or not to stream TF output in realtime

destroy(*stream=False*)
Run a 'terraform destroy'

Parameters **stream** (*bool*) – whether or not to stream TF output in realtime

plan(*stream=False*)
Run a 'terraform plan'

Parameters **stream** (*bool*) – whether or not to stream TF output in realtime

webhook2lambda2sqs.tf_generator module

webhook2lambda2sqs.utils module

webhook2lambda2sqs.utils.**pretty_json**(*obj*)
Given an object, return a pretty-printed JSON representation of it.

Parameters **obj** (*object*) – input object

Returns pretty-printed JSON representation

Return type str

webhook2lambda2sqs.utils.**read_json_file**(*fpath*)
Read a JSON file from `fpath`; raise an exception if it doesn't exist.

Parameters **fpath** (*str*) – path to file to read

Returns deserialized JSON

Return type dict

webhook2lambda2sqs.utils.**run_cmd**(*args*, *stream=False*, *shell=True*)
Execute a command via `subprocess.Popen`; return its output (string, combined STDOUT and STDERR) and exit code (int). If stream is True, also stream the output to STDOUT in realtime.

Parameters

- **args** – the command to run and arguments, as a list or string
- **stream** (*bool*) – whether or not to stream combined OUT and ERR in realtime
- **shell** (*bool*) – whether or not to execute the command through the shell

Returns 2-tuple of (combined output (str), return code (int))

Return type tuple

webhook2lambda2sqs.version module

Development

To install for development:

1. Fork the webhook2lambda2sqs repository on GitHub
2. Create a new branch off of master in your fork.

```
$ virtualenv webhook2lambda2sqs
$ cd webhook2lambda2sqs && source bin/activate
$ pip install -e git+git@github.com:YOURNAME/webhook2lambda2sqs.git@BRANCHNAME
↪#egg=webhook2lambda2sqs
$ cd src/webhook2lambda2sqs
```

The git clone you're now in will probably be checked out to a specific commit, so you may want to `git checkout BRANCHNAME`.

Guidelines

- pep8 compliant with some exceptions (see pytest.ini)
- 100% test coverage with pytest (with valid tests)

Testing

Testing is done via pytest, driven by tox.

- testing is as simple as:
  - `pip install tox`
  - `tox -e <environment name>`
- If you want to pass additional arguments to pytest, add them to the tox command line after "–". i.e., for verbose pytext output on py27 tests: `tox -e py27 -- -v`

Acceptance Tests

These will actually spin up the entire system end-to-end, send some messages via POST and GET, and test that they work. It *should* clean everything up when finished.

`tox -e acceptance`

Use `export NO_TEARDOWN=true` to prevent tear-down of the infrastructure. When you're ready to destroy it, `unset NO_TEARDOWN` and run `tox -e acceptance` again.

Release Checklist

1. Open an issue for the release; cut a branch off master for that issue.
2. Run the `acceptance` tox environment locally.
3. Confirm that there are CHANGES.rst entries for all major changes.
4. Ensure that Travis tests passing in all environments.
5. Ensure that test coverage is no less than the last release (ideally, 100%).
6. Increment the version number in webhook2lambda2sqs/version.py and add version and release date to CHANGES.rst, then push to GitHub.
7. Confirm that README.rst renders correctly on GitHub.
8. Upload package to testpypi:
   - Make sure your ~/.pypirc file is correct (a repo called `test` for https://testpypi.python.org/pypi)

    - `rm -Rf dist`
    - `python setup.py register -r https://testpypi.python.org/pypi`
    - `python setup.py sdist bdist_wheel`
    - `twine upload -r test dist/*`
    - Check that the README renders at https://testpypi.python.org/pypi/webhook2lambda2sqs
9. Create a pull request for the release to be merged into master. Upon successful Travis build, merge it.
10. Tag the release in Git, push tag to GitHub:
    - tag the release. for now the message is quite simple: `git tag -a X.Y.Z -m 'X.Y.Z released YYYY-MM-DD'`
    - push the tag to GitHub: `git push origin X.Y.Z`
11. Upload package to live pypi:
    - `twine upload dist/*`
12. make sure any GH issues fixed in the release were closed.

Changelog

0.2.0 (2017-06-25)

- Fix terraform output handling for 0.8.x
- Add Python 3.6 to test environment list
- Make compatible with Terraform 0.9+
- Add hack to `docs/source/conf.py` as workaround for https://github.com/sphinx-doc/sphinx/issues/3860

0.1.0 (2016-08-06)

- initial release

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

License

webhook2lambda2sqs is licensed under the GNU Affero General Public License, version 3 or later. This shouldn't be much of a concern to most people.

Python Module Index

W

Index

Symbols

A

C

D

E

F

G

I

L

P

R

S

T

W