mqttcloudproviderslib Documentation

Release 0.1.6

Costas Tyfoxylos

Jun 08, 2021

Contents

Contents:

CHAPTER 1

mqttcloudproviderslib

A library implementing cloud provider authentication for AWS, Azure and Google cloud for use in mqtt messaging and implements a message hub to publish messages in a fan pattern to all providers.

- Documentation: https://mqttcloudproviderslib.readthedocs.org/en/latest

1.1 Development Workflow

The workflow supports the following steps

- lint
- test
- build
- document
- upload
- graph

These actions are supported out of the box by the corresponding scripts under _CI/scripts directory with sane defaults based on best practices. Sourcing setup_aliases.ps1 for windows powershell or setup_aliases.sh in bash on Mac or Linux will provide with handy aliases for the shell of all those commands prepended with an underscore.

The bootstrap script creates a .venv directory inside the project directory hosting the virtual environment. It uses pipenv for that. It is called by all other scripts before they do anything. So one could simple start by calling _lint and that would set up everything before it tried to actually lint the project

Once the code is ready to be delivered the _tag script should be called accepting one of three arguments, patch, minor, major following the semantic versioning scheme. So for the initial delivery one would call

```
$ _tag –minor
```

which would bump the version of the project to 0.1.0 tag it in git and do a push and also ask for the change and automagically update HISTORY.rst with the version and the change provided.

So the full workflow after git is initialized is:

- repeat as necessary (of course it could be test - code - lint :))
  - code
  - lint
  - test
- commit and push
- develop more through the code-lint-test cycle
- tag (with the appropriate argument)
- build
- upload (if you want to host your package in pypi)
- document (of course this could be run at any point)

1.2 Important Information

This template is based on pipenv. In order to be compatible with requirements.txt so the actual created package can be used by any part of the existing python ecosystem some hacks were needed. So when building a package out of this **do not** simple call

```
$ python setup.py sdist bdist_egg
```

as this will produce an unusable artifact with files missing. Instead use the provided build and upload scripts that create all the necessary files in the artifact.

1.3 Project Features

- TODO

CHAPTER 2

Installation

At the command line:

```
$ pip install mqttcloudproviderslib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv mqttcloudproviderslib
$ pip install mqttcloudproviderslib
```

Or, if you are using pipenv:

```
$ pipenv install mqttcloudproviderslib
```

Or, if you are using pipx:

```
$ pipx install mqttcloudproviderslib
```

CHAPTER 3

Usage

To develop on mqttcloudproviderslib:

```
# The following commands require pipenv as a dependency

# To lint the project
_CI/scripts/lint.py

# To execute the testing
_CI/scripts/test.py

# To create a graph of the package and dependency tree
_CI/scripts/graph.py

# To build a package of the project under the directory "dist/"
_CI/scripts/build.py

# To see the package version
_CI/scripts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scripts/tag.py --major|--minor|--patch

# To upload the project to a pypi repo if user and password are properly provided
_CI/scripts/upload.py

# To build the documentation of the project
_CI/scripts/document.py
```

To use mqttcloudproviderslib in a project:

```
from mqttcloudproviderslib import Mqttcloudproviderslib
mqttcloudproviderslib = Mqttcloudproviderslib()
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Submit Feedback

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

4.1.1 Get Started!

Ready to contribute? Here's how to set up *mqttcloudproviderslib* for local development. Using of pipenv is highly recommended.

1. Clone your fork locally:

```
$ git clone https://github.com/schubergphilis/mqttcloudproviderslib
```

2. Install your local copy into a virtualenv. Assuming you have pipenv installed, this is how you set up your clone for local development:

```
$ cd mqttcloudproviderslib/
$ pipenv install --ignore-pipfile
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally. Do your development while using the CI capabilities and making sure the code passes lint, test, build and document stages.

4. Commit your changes and push your branch to the server:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

5. Submit a merge request

CHAPTER 5

mqttcloudproviderslib

5.1 mqttcloudproviderslib package

5.1.1 Submodules

5.1.2 mqttcloudproviderslib.mqttcloudproviderslib module

Main code for mqttcloudproviderslib.

class mqttcloudproviderslib.mqttcloudproviderslib.**AwsAdapter**(*device_name*, *endpoint*, *certificate*, *private_key*, *certificate_authority='AmazonRootCA1.pem'*, *port=443*, *protocol='x-amzn-mqtt-ca'*, *device_location='devices'*)

Bases: *mqttcloudproviderslib.mqttcloudproviderslib.BaseAdapter*

Placeholder.

on_disconnect(*client*, *user_data*, *return_code*)

Placeholder.

class mqttcloudproviderslib.mqttcloudproviderslib.**AzureAdapter**(*device_name*, *endpoint*, *key*, *api_version='2018-06-30'*, *certificate_authority='AzureRootCA.pem'*, *port=8883*, *protocol=4*)

Bases: *mqttcloudproviderslib.mqttcloudproviderslib.BaseAdapter*

Placeholder.

on_disconnect(*client*, *user_data*, *return_code*)
Placeholder.

class mqttcloudproviderslib.mqttcloudproviderslib.**BaseAdapter**(*device_name*, *port*, *certificate_authority*, *protocol*)
Bases: abc.ABC

Placeholder.

name
Placeholder.

on_disconnect(*client*, *user_data*, *return_code*)
Placeholder.

protocol
Placeholder.

publish(*message*)
Placeholder.

publish_to_subtopic(*message*, *topic*)
Placeholder.

class mqttcloudproviderslib.mqttcloudproviderslib.**GoogleAdapter**(*device_name*, *project_id*, *cloud_region*, *registry_id*, *mqtt_bridge_hostname*, *mqtt_bridge_port*, *private_key*, *certificate_authority='GoogleRoots.pem'*, *port=8883*, *protocol=<_SSLMethod.PROTOCOL_TLSv1_2: 5>*)
Bases: *mqttcloudproviderslib.mqttcloudproviderslib.BaseAdapter*

Placeholder.

on_disconnect(*client*, *user_data*, *return_code*)
Placeholder.

class mqttcloudproviderslib.mqttcloudproviderslib.**MessageHub**(*configuration*)
Bases: object

A fan provider to all cloud providers.

broadcast(*message*)
It will broadcast the provided message to all registered cloud provider's default topic.

Parameters **message** (*dict*) – The message to publish to the default provider's topic.

Returns True if all published messages get delivered, False if any fails.

Return type result (bool)

broadcast_to_subtopic(*message*, *topic*)
It will broadcast the provided message to all registered cloud provider's with specified topic.

Parameters

- **topic** (*str*) – The provider's specific topic to publish the message to.
- **message** (*dict*) – The message to publish to the specified provider's topic.

Returns True if all published messages get delivered, False if any fails.

Return type result (bool)

class mqttcloudproviderslib.mqttcloudproviderslib.**Provider**

Bases: object

Placeholder.

5.1.3 mqttcloudproviderslib.mqttcloudproviderslibexceptions module

Custom exception code for mqttcloudproviderslib.

exception mqttcloudproviderslib.mqttcloudproviderslibexceptions.**InvalidAzureKeyContents**

Bases: Exception

Could not read the provided file as a valid file holding an azure key.

exception mqttcloudproviderslib.mqttcloudproviderslibexceptions.**ProviderInstantiationError**

Bases: Exception

No provider could be instantiated because the data provided was invalid.

5.1.4 mqttcloudproviderslib.schemas module

Main code for mqttcloudproviderslib.

5.1.5 Module contents

mqttcloudproviderslib package.

Import all parts from mqttcloudproviderslib here

CHAPTER 6

Credits

6.1 Development Lead

- Costas Tyfoxylos <ctyfoxylos@schubergphilis.com>
- Frank Breedijk <fbreedijk@schubergphilis.com>

6.2 Contributors

- Marcel Bezemer <mbezemer@schubergphilis.com>

CHAPTER 7

History

CHAPTER 8

0.0.1 (03-02-2020)

- First code creation

CHAPTER 9

0.1.0 (04-02-2020)

- First MVP release.

CHAPTER 10

0.1.1 (04-02-2020)

- Updated cloud provider schemas with optional entries.

CHAPTER 11

0.1.2 (04-02-2020)

- Fixed package reference for relative dynamic import.

CHAPTER 12

0.1.3 (04-02-2020)

- Fixed Azure key contents and subtopic rendering.

CHAPTER 13

0.1.4 (05-02-2020)

- Fixed bugs with azure provider.

CHAPTER 14

0.1.5 (05-02-2020)

- Small linting fixes.

CHAPTER 15

0.1.6 (26-04-2021)

- Bumped dependencies.

CHAPTER 16

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index