secrets-helper

Release 0.1.0

Dec 11, 2019

Contributing

`secrets-helper` helps you use credentials stored in AWS Secrets Manager with tools that accept credentials and other configuration through environment variables.

CHAPTER 1

Getting Started

1.1 Required Prerequisites

- Suported Python versions
  - 3.7
  - 3.8

1.2 Installation

```
$ pip install secrets-helper
```

CHAPTER 2

Usage

2.1 How it works

To use `secrets-helper`, simply identify the ARN of your Secrets Manager secret, the command profile that you want to use, and the command that you want to run along with any desired arguments.

```
$ secrets-helper run \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret \
    --profile twine \
    --command "twine upload --skip-existing dist/*"
```

2.1.1 Value Injection

`secrets-helper` gets your credentials from Secrets Manager and injects them into the appropriate environment variables when it runs the command.

For example, if you were to have it run the `env` command to print out all of your environment variables, you would see:

```
$ secrets-helper run \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret \
    --profile twine \
    --command env
TWINE_USERNAME=my-awesome-username
TWINE_PASSWORD=my-secret-password
TWINE_REPOSITORY_URL=https://test.pypi.org/legacy/
```

Important: Storing secrets in environment variables carries risks because anything running in that environment has access to those secret values.

For example, if your testing infrastructure logs all environment variables (something that is perfectly reasonable to do in a testing environment) then if you store secrets in your environment variables those secrets are leaked into your testing logs.

`secrets-helper run` helps you work around this by running your requested command in an isolated subprocess. Your secret environment variables are ONLY injected into this isolated subprocess, so anything running outside of that process does NOT have access to your secrets.

2.1.2 Configuration

In order for `secrets-helper` to correctly load your credentials, your secret needs to be JSON-formatted with expected field names. It uses these names to map your secret values to the correct environment variables. This means you can use a single Secrets Manager secret to store multiple secret values.

You can either format your secret plaintext like this manually or you can create key-value pairs in your secret through the Secrets Manager console.

```
$ aws secretsmanager get-secret-value \
    --secret-id arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret
{
    "username": "my-awesome-username",
    "password": "my-secret-password",
    "url": "https://test.pypi.org/legacy/"
}
```

`secrets-helper` comes pre-loaded with some known environment variable mappings for common tools. You can use those as-is or you can provide an ini-style config file that defines a custom mapping.

```
$ secrets-helper run \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret \
    --config twine-config.cfg \
    --command "twine upload --skip-existing dist/*"
```

A mapping for `twine` might looks like this:

```
[secrets-helper.env]
username: TWINE_USERNAME
password: TWINE_PASSWORD
url: TWINE_REPOSITORY_URL
```

Warning: Each identifier to environment variable mapping MUST be 1:1. No environment variable may have more than one identifier that maps to it.

2.1.3 Additional Configuration

You can also define secret IDs and a command profile in the config file.

```
[secrets-helper.settings]
secrets:
    arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret
    arn:aws:secretsmanager:us-west-2:111222333444:secret:AnotherSecret
profile: twine
```

2.2 Multiple Secrets

You might need to load secret values from multiple Secrets Manager secrets. In this case, simply provide multiple `--secret` options!

Warning: WARNING: If you use `secrets-helper` with multiple secrets, those secrets MUST NOT have any repeating keys.

```
$ secrets-helper run \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:AnotherSecret \
    --profile twine \
    --command "twine upload --skip-existing dist/*"
```

2.3 Secrets as Command Line Parameters

Some commands need secret values to be passed in as command line parameters. If you need to do this, add a reference in your `--command` parameter using the `{env:NAME}` syntax to identify where you need environment variable values to be injected.

Important: Providing secrets as command line arguments should generally be avoided because anything monitoring your shell activity (such as `history`) will collect your secret values.

`secrets-helper run` helps with this too! Because it executes the command you request in an isolated subprocess, the command arguments are not exposed to external processes.

```
$ secrets-helper run \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret \
    --profile twine \
    --command "twine upload --skip-existing {env:DIST_DIRECTORY}"
```

2.4 Passing to `env`

In some advanced use cases, you might not be able to use the `run` operating mode. If you find yourself in this situation, you can use the `env` operating mode and make `secrets-helper` print out the resulting environment variable assignments in a format that the `env` command can understand.

```
$ secrets-helper env \
    --secret arn:aws:secretsmanager:us-west-2:111222333444:secret:MyAwesomeSecret \
    --profile twine
```

CHAPTER 3

Development

3.1 Prerequisites

- Required
  - Python 3.7+
  - tox : We use tox to drive all of our testing and package management behavior. Any tests that you want to run should be run using tox.
- Optional
  - pyenv : If you want to test against multiple versions of Python and are on Linux or MacOS, we recommend using pyenv to manage your Python runtimes.
  - tox-pyenv : Plugin for tox that enables it to use pyenv runtimes.
  - detox : Parallel plugin for tox. Useful for running a lot of test environments quickly.

3.1.1 Setting up pyenv

If you are using pyenv, make sure that you have set up all desired runtimes and configured the environment before attempting to run any tests.

1. Install all desired runtimes.
   - ex: `pyenv install 3.7.0`
   - **NOTE:** You can only install one runtime at a time with the `pyenv install` command.
1. In the root of the checked out repository for this package, set the runtimes that pyenv should use.
   - ex: `pyenv local 2.7.14 3.4.6 3.5.3 3.6.4 3.7.0`
   - **NOTE:** This creates the `.python-version` file that pyenv will use. Pyenv treats the first version in that file as the default Python version.

3.2 Running tests

There are two criteria to consider when running our tests: what version of Python do you want to use and what type of tests do you want to run?

For a full listing of the available types of tests available, see the `[testenv]commands` section of the `tox.ini` file.

All tests should be run using tox. To do this, identify the test environment that you want tox to run using the `-e ENV_NAME` flag. The standard test environments are named as a combination of the Python version and the test type in the form `VERSION-TYPE`. For example, to run the `local` tests against CPython 3.7:

```
tox -e py37-local
```

If you want to provide custom parameters to pytest to manually identify what tests you want to run, use the `manual` test type. Any arguments you want to pass to pytest must follow the `--` argument. Anything before that argument is passed to tox. Everything after that argument is passed to pytest.

```
tox -e py37-manual -- test/unit/test_example_file.py
```

3.3 Before submitting a pull request

Before submitting a pull request, please run the `lint` tox environment. This will ensure that your submission meets our code formatting requirements and will pass our continous integration code formatting tests.

3.3.1 Internal Resources

Warning: These are provided for informational purposes only. No guarantee is provided on the modules and APIs described here remaining consistent. Directly reference at your own risk.

`secrets_manager_helper.internal`

CHAPTER 4

Changelog

4.1 0.1.0 – 2019-12-11

Initial release.