awsauthhelper Documentation

Release 1.4.1

Drew J. Sonne

April 05, 2016

Contents

Helper library providing ArgumentParser and Credentials class for AWS authentication

CHAPTER 1

Introduction

A Python library for simplifying authentication to Amazon Web Services APIs.

Provides support for:

Authentication:

- one
- two

Password generation

- one

1.1 Changes

For details on the latest updates and changes, see What's new in aws-auth-helper 1.4.1

1.2 License

This software is released under the GPLv2.

See the `licence` for full text.

1.3 Dependencies

- Python 2.7.10

1.4 Installation

See Installing aws-auth-helper for details.

1.5 Documentation

This library has comprehensive docstrings and a full set of project documentation (including tutorials):

- http://pythonhosted.org/aws-auth-helper/
- http://readthedocs.org/docs/aws-auth-helper/en/latest/

1.6 Finally...

Share and enjoy!

CHAPTER 2

Installing aws-auth-helper

aws-auth-helper is available in various packaged and non-packaged forms :

- source code repository access
- source release packages (tarball and zip formats)
- Python eggs

You can also build your own RPM packages, using bdist_rpm with setup.py available in the source tarball.

2.1 Locating the software

aws-auth-helper is available directly from the public subversion source code repository.

Details on how to check out the source code can be found here :

http://github.com/drewsonne/aws-auth-helper/

Official milestone releases can be found here :

http://github.com/drewsonne/aws-auth-helper/downloads

2.2 Source Release Packages

Download the latest release tarball/zip file and extract it to a temporary location or check out the source from the code hosting site into a local working copy directory.

Run the setup file in the root directory like this:

```
python setup.py install
```

This automatically places the required files in the `lib/site-packages` directory of the Python version you used to run the setup script, may be part of a virtualenv or similar.

2.3 Python Eggs

You can build and install eggs with aws-auth-helper using the `setup_egg.py` file provided in the source distribution.

All the usual commands are supported e.g.:

```
python setup_egg.py develop
python setup_egg.py bdist_egg
...
```

This requires that you install distribute or setuptools which is not part of the Python standard library.

See the following URL for details :-

- `distribute` - http://guide.python-distribute.org/
- `setuptools` (old) - http://peak.telecommunity.com/DevCenter/setuptools

Warning: `setuptools` is now very long in the tooth and full of bugs! Just use distribute, or pip instead.

Download and install the latest easy_install script and run the following command

```
easy_install aws-auth-helper
```

This will go to the Python Package Index and automatically find the appropriate version of aws-auth-helper for your Python setup.

Alternatively, you can use pip instead of easy_install.

Just download the latest version of pip from PyPI found here - http://pypi.python.org/pypi/pip and run the following command

```
pip install aws-auth-helper
```

2.4 Final Words

Always be sure you verify your downloads against the checksums on the code hosting site's download page!

CHAPTER 3

Tutorial

This tutorial will take you through a quick example of a normal ArgumentParser class, and then show you how to integrate the AWSArgumentParser into this.

First import the default ArgumentParser

```
>>> from argparse import ArgumentParser
```

First, let's create an argument parser for the rest of the options in our new utility.

```
>>> # Instantiate an argument parser, add an argument, and print the help text
>>> my_aws_app = ArgumentParser(description='Lists EC2 instances', prog='my_app')
>>> my_aws_app.add_argument('--name', required=True)
>>> my_aws_app.print_help()
usage: my_app [-h] --name NAME

Lists EC2 instances

optional arguments:
  -h, --help          show this help message and exit
  --name NAME

>>> my_aws_app.parse_args(args=['--name', 'Hello, World!'])
Namespace(name='Hello, World!')
```

Now that we have a parser for the arguments of our utility, we can add the AWSArgumentParser.

```
>>> from awsauthhelper import AWSArgumentParser
>>> aws_options = AWSArgumentParser(role_session_name='ec2_audit')
```

Note: You **must** set a *role_session_name* parameter in-case the user does not provide one on the cli.

Now let's recreate our app options, so that we can chain the AWSArgumentParser.

```
>>> my_aws_app = argparse.ArgumentParser(
>>>     prog='my_app',
>>>     description='Lists EC2 instances',
>>>     parents=[
>>>       aws_options
>>>     ]
>>> )
>>> my_aws_app.add_argument('--max-instances', type=int)
>>> my_aws_app.print_help()
```

```
usage: my_app [-h] [--aws-access-key-id AWS_ACCESS_KEY_ID]
              [--aws-secret-access-key AWS_SECRET_ACCESS_KEY]
              [--aws-session-token AWS_SESSION_TOKEN] [--region REGION]
              [--profile PROFILE] [--role ROLE] [--config-path CONFIG_PATH]
              [--credentials-path CREDENTIALS_PATH] [--auth-debug]
              [--role-session-name ROLE_SESSION_NAME]
              [--max-instances MAX_INSTANCES]

Lists EC2 instances

optional arguments:
  -h, --help            show this help message and exit
  --max-instances MAX_INSTANCES

AWS credentials:
  --aws-access-key-id AWS_ACCESS_KEY_ID
                        AWS access key
  --aws-secret-access-key AWS_SECRET_ACCESS_KEY
                        Access and secret key variables override credentials
                        stored in credential and config files
  --aws-session-token AWS_SESSION_TOKEN
                        A session token is only required if you are using
                        temporary security credentials.
  --region REGION       This variable overrides the default region of the in-
                        use profile, if set.
  --profile PROFILE     This can be the name of a profile stored in a
                        credential or config file, or default to use the
                        default profile.
  --role ROLE           Fully qualified role arn to assume
  --config-path CONFIG_PATH
                        Specify a custom location for ~/.aws/config
  --credentials-path CREDENTIALS_PATH
                        Specify a custom location for ~/.aws/credentials
  --auth-debug          Enter debug mode, which will print credentials and
                        then exist at `create_session`.
  --role-session-name ROLE_SESSION_NAME
                        If you have assigned a role, set a --role-session-name
```

Note: It is possible to use the AWSArgumentParser as your main ArgumentParser object, and like you would with a normal ArgumentParser object, but if you chain the ArgumentParser and AWSArgumentParser, you can segment your options in the help text, as you can see here. Furthermore, if you set the AWSArgumentParser as the parent, the aws options will be rendered at the end of the help.

CHAPTER 4

API Reference

4.1 AWSArgumentParser

class awsauthhelper.**AWSArgumentParser**(*role_session_name*, *region=None*, *profile=None*, *enforce_auth_type=None*, ***kwargs*)

Helper Class containing a preset set of cli arguments for parsing into the Credentials object. If not explicity set, arguments are read from the environment variables.

Create our arguments and determine if we need to enforce an auth method.

Parameters

- **role_session_name** (*str*) – Default name for the role session, in case a user does not provide one.
- **region** (*str*) – AWS Region
- **profile** (*str*) – Name of the profile in the AWS profile to use as the base configuration.
- **enforce_auth_type** (*str*) – The Authentication method can be locked to one of {'keys', 'keys_with_session', 'profile', 'profile_role','config','credentials'}
- **kwargs** (*dict*) –

Return awsauthhelper.AWSArgumentParser

This class provides a prepackaged set of cli options for AWS authentication.

CLI Option	Default	Description
--aws-access-key	$AWS_ACCESS_KEY_ID	
--aws-secret-access-key	$AWS_SECRET_ACCESS_KEY	
--aws-session-token	$AWS_SESSION_TOKEN	
--region	$AWS_DEFAULT_REGION	
--profile	$AWS_DEFAULT_PROFILE	
--role		
--config-path	$AWS_CONFIG_FILE	Custom path to an AWS config file
--credentials-path	$AWS_SHARED_CREDENTIALS_FILE	Custom path to an AWS credentials path
--auth-debug		If this flag is enabled, execution of the application will stop when create_session() is called.

The *AWSArgumentParser* class takes all the arguments of a argparser.ArgumentParser class in addition to:

- *role_session_name* is a default value in case --role_session_name is not provided by the user.
- *region* is a default value in case --region is not provided by the user.

- *profile* is a default value in case `--profile` is not provided by the user.
- *enforce_auth_type* enforces the type of arguments which must be passed to this utility. Can be one of:

Like `argparse.ArgumentParser`, *AWSArgumentParser* allows chaining/inclusion of multiple `ArgumentParser` objects through the `list[argparse.ArgumentParser]: parents` constructor argument. The child `ArgumentParser` appears last in the list of options when `--help` is called, so it's best to add *other* `ArgumentParser` objects to *AWSArgumentParser*, rather than the reverse.

4.2 validate_creds

Helper function validate your credential combinations

awsauthhelper.**validate_creds**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *aws_session_token=None*, *profile=None*, ***kwargs*)

Perform validation on CLI options

Parameters

- **aws_access_key_id** (*str*) –
- **aws_secret_access_key** (*str*) –
- **aws_session_token** (*str*) –
- **profile** (*str*) –
- **kwargs** –

Raises

- **argparse.ArgumentError** – If `--aws-session-token` is specified but `--aws-secret-access-key` and `--aws-access-key-id` are not
- **argparse.ArgumentError** – If `--profile` is specified and `--aws-secret-access-key` or `--aws-access-key-id` are also specified.
- **argparse.ArgumentError** – If one of `--aws-secret-access-key` or `--aws-access-key-id` have been provided but not both.

Returns

4.3 Credentials

class awsauthhelper.**Credentials**(*region=None*, *aws_secret_access_key=None*, *aws_access_key_id=None*, *aws_session_token=None*, *profile=None*, *role=None*, *role_session_name=None*, *config_path=None*, *credentials_path=None*, *auth_debug=False*, ***kwargs*)

Encapsulates processing of AWS credentials.

Handle the assumption of roles, and creation of Session objects.

Parameters

- **region** (*str*) – AWS region

- **aws_secret_access_key** (*str*) – AWS_SECRET_ACCESS_KEY to use for the base credentials.
- **aws_access_key_id** (*str*) – AWS_ACCESS_KEY_ID to use for the base credentials.
- **aws_session_token** (*str*) – AWS_SESSION_TOKEN to use for the base credentials. Generally this should not be needed as roles are assumed through providing a role argument.
- **profile** (*str*) – Name of the profile in the AWS profile to use as the base configuration.
- **role** (*str*) – ARN of the AWS IAM Role to assume.
- **role_session_name** (*str*) – Custom name of the role session to override the default.
- **config_path** (*str*) – Custom path to the aws config file if it is not in a location botocore expects.
- **credentials_path** (*str*) – Custom path to the aws credentials file if it is not in a path botocore expects.
- **auth_debug** (*bool*) – Whether or not to print debug information. If True, exit() is throw at create_session()
- **kwargs** (*dict*) – catcher to allow arbitrary **var(my_args.parse_args(...)) to be passed in. Arguments in **kwargs not used at all.

Return awsauthhelper.Credentials

members

4.4 Password generation

awsauthhelper.password.**generate**(*password_policy*)

Builds a password based on the password policy provided password_policy should be an object with the attributes:

- **minimum_password_length** *(int)* – Minimum length of password. Maximum length of password will be the ceiling of 1.3 times this value.
- **require_symbols** *(bool)* – Make sure password contains !@#$%^&*()_+-=[]{}|'.
- **require_lowercase_characters** *(bool)* – Make sure password contains abcdefghijklmnopqrstuvwxyz.
- **require_uppercase_characters** *(bool)* – Make sure password contains ABCDEFGHIJKLMNOPQRSTUVWXYZ.
- **require_numbers** *(bool)* – Make sure password contains 0123456789.

Parameters **password_policy** (*iam.AccountPasswordPolicy*) – boto password policy

Return basestring New password

CHAPTER 5

What's new in aws-auth-helper 1.4.1

5.1 Release: 1.4.0

Date: 4 Sep 2015

5.1.1 Changes since 1.3.3

- Added method Credentials.use_as_global() to allow the use of credentials with calls directly in the boto3 namespace.
- Added more test cases
- Changed default, profile, region, and role_session name parameter names in awsauthhelper.Credentials.__init__

5.1.2 Specific bug fixes addressed in this release

FIXED 10: **https://github.com/drewsonne/aws-auth-helper/issues/10**

- __init__.py - Bad key

CHAPTER 6

Indices and tables

- genindex
- search

A

C

G

V