S3Vault Library Documentation

Release 3.1.0

Giuseppe Chiesa

Jan 09, 2020

Contents

Contents:

CHAPTER 1

S3Vaultlib

S3Vaultlib is a Python Library and CLI tool that enable you to implements a secure vault / configuration datastore for your AWS platform by using AWS resources: CloudFormation, S3, IAM, KMS S3Vaultlib it's **yet another vault** with the goal to give easy maintainability, use only AWS resource and with strong security patterns in mind.

1.1 Why a vault?

It's a common pattern in SRE and DevSecOps to create resources environment unaware and configure the resource automatically when is deployed in a specific environment

1.2 S3Vaultlib Features

- Use Server Side Encryption to store the objects on S3 with per-role KMS key
- Use per role encryption with least privilege patterns to access the vault. Each role in the vault **can only consume** its own keymaterials
- Special elevated privileged mode with a specific role able to produce and configure keymaterials, with only temporary access
- Save, retrieve, update objects in the vault
- Integrates flawlessly with Ansible by exposing an action plugin that allows you to expand templates by using variables / keymaterials from the vault
- Powerful CLI to create, manage and update the objects in the vault
- Easy maintainable via simple yaml file
- Expose a flexyble python library to extend functionalities or implement the retrieval of keymaterials from your code.

1.3 S3vaultlib Architecture

S3Vaultlib requires no installation or security patches / updates. The architecture leverages entirely on AWS existing resource to create a secure vault with Role Base Access Control, versioning and region awareness.

It integrates with the **IAM** to generate the necessary roles and policies, **KMS** to generate per-role keys, **S3** to configure the bucket policies to enforce high level of security and **CloudFormation** to create the Infrastructure as Code that combine all the above in a powerful vault.

Check In depth Architecture for more information

1.4 HOW-TOs

1.4.1 Example scenarios

- Provisioning a vault: A simple example to see how to provision a vault via the command line interface
- Configure NGINX with S3Vaultlib: A simple example where we deploy an environment unaware NGINX instance and it's configured via S3Vaultlib ansible plugin

1.4.2 CLI Usage

The complete documentation can be found here: CLI Usage

1.5 Alternatives

Currently there are several alternative patterns used.

- Configuration / Keymaterials encrypted in git
  Please don't do this, really!
- Vault by Hashicorp
  Full featured vault system, widely used in the DevOPS community. But it's also yet another system to deploy and maintain in high availability and also, it requires keymaterials for the installation (since is not a native AWS component)
- AWS Secret Manager
  Very valid alternative offered by AWS. Still lack a bit of flexibility to be used transparently in your bootstrap pipelines for EC2 / Dockers / Lambdas / Applications

CHAPTER 2

Library Installation

2.1 Stable release

To install S3Vault Library, run this command in your terminal:

```
$ pip install s3vaultlib
```

This is the preferred method to install S3Vault Library, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for S3Vault Library can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/s3vaultlib/s3vaultlib
```

Or download the tarball:

```
$ curl  -OL https://github.com/s3vaultlib/s3vaultlib/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Library Usage

To use S3Vault Library in a project:

```
import s3vaultlib
```

3.1 Creating a S3Vault

You need a proper configuration file that describe your S3Vault setup (you can check `resources/s3vault.example.yml` as reference:

```
from s3vaultlib.configmanager import ConfigManager
from s3vaultlib.policymanager import PolicyManager

config = ConfigManager('s3vaultlib/resources/s3vault.example.yml')
policies = PolicyManager(config)
cloudformation_template = policies.generate_cloudformation()
```

You can then apply the cloudformation template to your AWS account and it will take care to configure the bucket to host an S3Vault

3.2 Managing a s3vault

- Instantiate a vault:

```
import s3vaultlib
from s3vaultlib.connectionfactory import ConnectionFactory
conn_manager = ConnectionFactory()
s3vault = S3Vault('my-bucket', '/vault', connection_factory=conn_manager)
```

- Upload a file in the vault:

```
# the encryption key will be guessed by resolving a KMS-Alias with the name of the␣
↪role of the EC2 instance
# where you are running the script
metadata = s3vault.put_file(src='test.dat', dest='test.dat')
```

- Update a configuration file in the vault:

```
# explicit usage of KMS-Alias
s3vault.set_property(configfile='myconfiguration', key='username', value='test_user',␣
↪key_alias='my-kms-alias')
```

- Expand a template file from a S3Vault

Assuming there is a object in the vault named `mycert` we can create a template like the following:

```
$ cat mycert.tpl
{{ mycert }}
```

and we can expand the template with the library:

```
rendered_data = s3vault.render_template('mycert.tpl')
print(rendered_data)
```

3.3 Extended documentation

Check out the module autogenerated documentation here: modindex

CHAPTER 4

Command Line Interface: s3vaultcli

S3Vaultlib ships also a powerful command line interface to interact with several functionalities

4.1 General Help

To check the latest version of the features and command available the inline help is the main reference

```
s3vaultcli --help
```

for each subcommand you can get detailed help with:

```
s3vaultcli <command> --help
```

4.2 Vault Provisioning

4.2.1 Create S3Vault Configuration

This command creates an example of the YAML configuration that is the starting point to provision a Vault

example:

```
s3vaultcli create_s3vault_config --output vault.yml
```

4.2.2 Create Cloudformation

This command generate the cloudformation based on the Vault YAML configuration.

example:

```
s3vaultcli create_cloudformation --config vault.yml --output vault.template
```

4.3 Vault objects management

4.3.1 Push

Upload a object in the Vault

example:

```
s3vaultcli push -b my_bucket_example -p webserver -k role_webserver -s mycert_key -d␣
↪mycert_key
```

NOTE: *please notice that S3Vaultlib does not support dots(.) in the objects to push to the vault*

4.3.2 Get

Retrieve a object from the Vault

example:

```
s3vaultcli get -b my_bucket_example -p webserver -k role_webserver -s mycert_key -d␣
↪mycert_key
```

example: from an EC2 instance with the role **role_webserver** associated

```
s3vaultcli get -b my_bucket_example -p webserver -s mycert_key -d mycert_key
```

NOTE: *if there is a role associated in the instance where the s3vaultcli perform a call, S3Vaultlib will try to detect the role name and then use the alias with the same name as the role*

4.3.3 Configuration Set

Create or update a configuration object in the Vault

example:

```
s3vaultcli configset -b my_bucket_example -p webserver -k role_webserver -c conf_
↪nginx -K server_name -V www.example.com
```

S3Vaultcli can also create more complex objects and hierarchies. Like the following example:

example: create a list object with the key `routed_network` inside the configuration object conf_vpn

```
s3vaultcli configset -b my_bucket_example -p webserver -k role_webserver -c conf_vpn -
↪K routed_networks -V '192.168.10.0/24, 192.168.11.0/24' -T list
```

S3Vaultcli can also attach a JSON or YAML object directly as subkey

example: create a sub object with the content of the YAML file `data.yml` inside the configuration object conf_vpn

```
s3vaultcli configset -b my_bucket_example -p webserver -k role_webserver -c conf_vpn -
↪K routed_networks -V data.yml -T yaml
```

4.3.4 Configuration Edit

This command will open a configuration editor inline (and in memory only) to dynamically view/change the content of a configuration object. The editor is quite powerful, supports **realtime validation** of the format (JSON/YAML) and **syntax highlighting**.

example: edit the configuration for the `conf_vpn` object as YAML file in memory

```
s3vaultcli configedit -b my_bucket_example -p webserver -k role_webserver -c conf_vpn␣
↪-t yaml
```

4.4 Template Expansion

4.4.1 Template

This command parse a Jinja2 template file and expands the jinja2 variables by retriving the information from the Vault

example:

```
s3vaultcli template -b my_bucket_example -p webserver -k role_webserver -t template.
↪j2 -d output.txt
```

NOTE: for more example see the *Configure NGINX with S3Vaultlib Ansible Plugin*

4.4.2 Ansible support

In order to be able to use / load the plugin for ansible you should export the ansible role shipped with s3vaultlib in the role_path in ansible:

example:

```
s3vaultcli ansible_path
```

CHAPTER 5

How-Tos

Contents:

5.1 Provisioning a vault

This document describe how to provision a vault using the S3Vaultlib CLI and AWS CloudFormation (via the console)

5.1.1 Provisioning the S3Vault

Let's generate the default config with:

```
s3vaultcli create_s3vault_config -o myconfig.yml
```

- edit the example configuration by setting the target S3 bucket to use as vault. The output should look like the following (comments are stripped out):

```
---
s3vaultlib:
  vault:
    bucket: "test-bucket-for-s3-vault"
  roles:
    - name: role_admin
      privileges: [read, write]
      path: _all_
      managed_policies: []
    - name: role_webserver
      privileges: [read]
      kms_alias: role_webserver
      path:
        - webserver/
    - name: role_mysql
      privileges: [read]
```

(continues on next page)

(continued from previous page)

```
      kms_alias: role_mysql
      path:
      - mysql/
```

- with the CLI now we are going to create the cloudformation template for the vault

```
s3vaultcli create_cloudformation -c test.yml -o test.template
```

- Now in the AWS Console we enter CloudFormation and we create a new template from file and we upload `test.template`. In a while we should have our vault completely configured.

5.2 Configure NGINX with S3Vaultlib Ansible Plugin

Let's imagine a simple scenario where we need to deploy a nginx instance. The instance is environment unaware and only at deploy time we need to provision a server name and a port dynamically, and a `htpasswd` file for basic authentication.

Let's also assume that the EC2 instance running NGINX will have the `role_webserver` associated (see *Provisioning a vault*)

5.2.1 Provisioning of keymaterials

- We need to create a session as `role_admin` in order to provision the keymaterials and configuration

```
s3vaultcli create_session -r role_admin --no-external-id
```

- With the CLI we can provision a configuration file that holds the settings we need:

```
s3vaultcli configset -b 'test-bucket-for-s3-vault' -p webserver -k role_webserver -c␣
↪conf_nginx -K server_name -V www.example.com
s3vaultcli configset -b 'test-bucket-for-s3-vault' -p webserver -k role_webserver -c␣
↪conf_nginx -K server_port -V 8443
```

- Now we push as separate keymaterial the `htpasswd` (prebuilt) file

```
s3vaultcli push -b 'test-bucket-for-s3-vault' -p webserver -k role_webserver -s␣
↪htpasswd -d htpasswd
```

NOTE*: if you don't pass the kms_alias, the library will try to detect the role and use a KMS key with the same alias of the role name. If we are in another machine (or from our local machine we need to have access to the KMS key and specify the alias with the -k key_alias option)*

5.2.2 What's now on S3 vault?

Let's just stop a moment and see how the library actually implements the vault. If we check S3 from AWS Console as privileged user (with S3 read privileges) or with the aws CLI we will notice a structure like the following:

```
$ aws s3 ls s3://test-bucket-for-s3-vault/webserver/
2017-08-20 18:02:10          5 htpasswd
2017-08-20 18:00:39         57 conf_nginx
```

- `htpasswd` has been saved as binary file via the push method

- `conf_nginx` is a container (JSON file) that holds all the key and values set via the CLI. The JSON container will allow you to create very complex structure and retrieve them at runtime.

5.2.3 Prepare the configuration on EC2

NGINX Conf

Now we can prepare the templates to expand when the instance starts. Best practice is to preallocate the templates in an EC2 AMI using Packer by Hashicorp. Let's create the file `/opt/templates/nginx.conf.j2` with the content:

```
server {
        listen {{ conf_nginx.server_port }} default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name {{ conf_nging.server_name }};
        location / {
          root   html;
          index  index.html index.htm;
        }
}
```

htpasswd

Now inside the instance we can prepare the templates to expand when the instance starts. Let's create the file `/opt/templates/htpasswd.j2` with the content:

```
{{ htpasswd }}
```

5.2.4 EC2 Startup

Now we can launch an EC2 instance, keeping in mind the followings:

- We need to make sure the EC2 instance will use the pre-baked AMI generated with Packer
- We need to associate to the EC2 instance the role `role_webserver`
- We need to setup the following command (beside the rest) in userdata

```
s3vaultcli template -b 'test-bucket-for-s3-vault' -p webserver -t /opt/nginx.conf.j2 -
↪d /etc/nginx/nginx.conf
s3vaultcli template -b 'test-bucket-for-s3-vault' -p webserver -t htpasswd.j2 -d /etc/
↪nginx/htpasswd
chown nginx /etc/nginx/htpasswd && chmod go-rwx /etc/nginx/htpasswd
```

5.2.5 Ansible Support

Instead using `s3vaultcli template` we can also automate the provisioning of keymaterials via the Ansible Action Plugin shipped together with the library. The Ansible Plugin expose a new command `s3vault_template`. The command has the same capabilities of the `template` command in Ansible with the additional feature that all the variables in the template are resolved using the Vault.

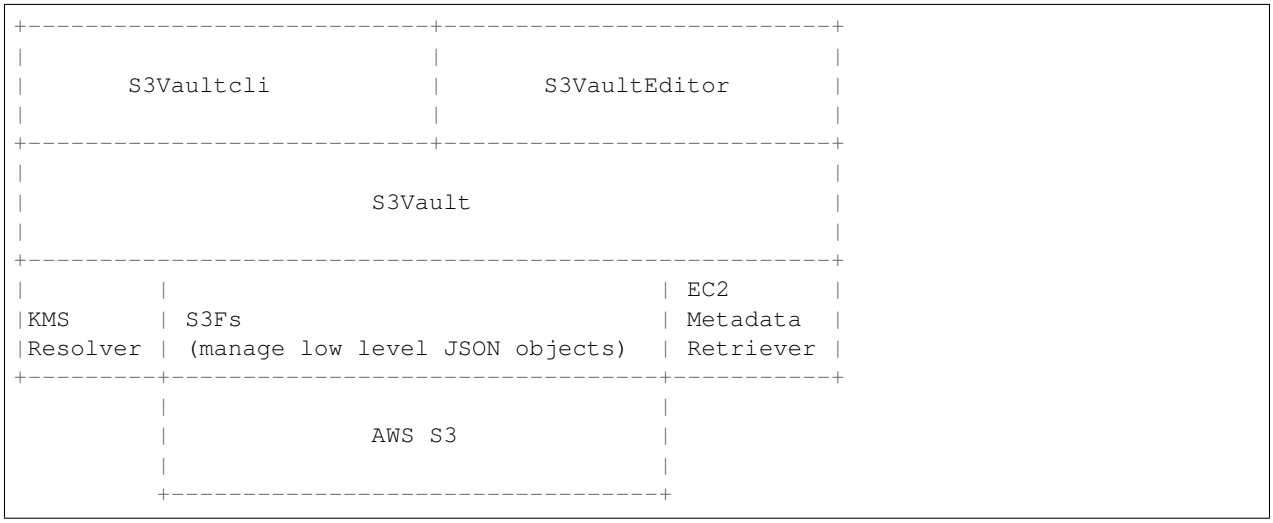
Example:

```
- name: Set nginx configuration
  s3vault_template:
    bucket: test-bucket-for-s3-vault
    path: webserver
    src: /opt/nginx.conf.j2
    dest: /etc/nginx/nginx.conf
    mode: 0600
    owner: nginx
    group: nginx
```

CHAPTER 6

Architecture

The following diagram show the overall architecture of the S3Vaultlib



- **S3Fs** is the layer that abstracts the object management on S3 with ServerSide Encryption
- **KMS Resolver** is an helper that resolves the KMS Key from the role of from aliases
- **EC2 Metadata Retriever** is an helper that tries to lookup the role when the tool is used inside a AWS resource
- **S3Vault** is the main library that export high level APIs to be used in other applications
- **S3Vaultcli** is the Command Line Interface
- **S3VaultEditor** is the inline and in memory editor to change the content of the vault objects

CHAPTER 7

s3vaultlib

7.1 s3vaultlib package

7.1.1 Subpackages

s3vaultlib.cloudformation package

Submodules

s3vaultlib.cloudformation.policymanager module

class s3vaultlib.cloudformation.policymanager.**PolicyManager**(*config_manager*)
Bases: object

generate_cloudformation()

get_policy_variables()

with_account_id(*account_id*)

with_connection_factory(*connection_factory*)

s3vaultlib.cloudformation.policymanager.**cloudformation_prettyprint**(*json_string*)

s3vaultlib.cloudformation.policymanager.**cloudformation_sanitize_string**(*string*)

Module contents

s3vaultlib.config package

Submodules

s3vaultlib.config.configmanager module

exception s3vaultlib.config.configmanager.**ConfigException**
Bases: exceptions.Exception

class s3vaultlib.config.configmanager.**ConfigManager**(*config_file*)
Bases: object

get_all_path()

load_config()

load_roles(*s3vault_configdata=None*)

load_vault(*s3vault_configdata*)

parse_role_config(*role_config*)

path_all

s3vaultlib.config.role module

class s3vaultlib.config.role.**Role**(*name*, *config_obj*)
Bases: object

SUPPORTED_PRIVILEGES = ('read', 'write')

is_path_all()

kms_alias

kms_arn

managed_policies

name

path

privileges

with_connection_factory(*connection_factory*)

exception s3vaultlib.config.role.**RoleException**
Bases: exceptions.Exception

Module contents

s3vaultlib.connection package

Submodules

s3vaultlib.connection.connectionmanager module

class s3vaultlib.connection.connectionmanager.**ConnectionManager**(*region=None*, *endpoint=None*, *is_ec2=False*, ***params*)
Bases: object

Object that allocate connection by supporting also connection profile and extended paramaters

client (*resource*)
Returns a client connection

is_ec2

s3vaultlib.connection.defaults module

s3vaultlib.connection.tokenmanager module

class s3vaultlib.connection.tokenmanager.**TokenManager**(*role_name=None*, *role_arn=None*, *external_id=None*, *connection_factory=None*, *is_ec2=False*)

Bases: object

TOKEN_FILENAME = '~/.s3vaultlib.token'

generate_token()

has_token()

static remaining_seconds(*token_dict*)

role_arn

token

exception s3vaultlib.connection.tokenmanager.**TokenManagerException**
Bases: exceptions.Exception

Module contents

s3vaultlib.editor package

Submodules

s3vaultlib.editor.autosuggestions module

class s3vaultlib.editor.autosuggestions.**AutosuggestFromDocumentData**(*bottom_toolbar_attributes=None*, *mode=u'json'*, ***kwargs*)

Bases: prompt_toolkit.auto_suggest.AutoSuggest

get_suggestion(*buffer*, *document*)

s3vaultlib.editor.completers module

class s3vaultlib.editor.completers.**CompleteFromDocumentKeys**(*bottom_toolbar_attributes=None*, *mode=u'json'*, ***kwargs*)

Bases: prompt_toolkit.completion.base.Completer

get_completions(*document*, *complete_event*)

s3vaultlib.editor.editor module

class s3vaultlib.editor.editor.**Editor**(*json_data*, *attributes=None*, *mode=u'json'*)
Bases: object

LEXERS = {u'json': <class 'pygments.lexers.data.JsonLexer'>, u'yaml': <class 'pygmen

SUPPORTED_MODE = (u'json', u'yaml')

VALIDATORS = {u'json': <class 's3vaultlib.editor.validators.JSONValidator'>, u'yaml':

bottom_bar()

data

lexer_class

result

run()

validator_class

exception s3vaultlib.editor.editor.**EditorAbortException**
Bases: exceptions.Exception

exception s3vaultlib.editor.editor.**EditorException**
Bases: exceptions.Exception

class s3vaultlib.editor.editor.**MessageDialog**(*title*, *text*)
Bases: object

s3vaultlib.editor.utils module

s3vaultlib.editor.utils.**extract_tokens**(*dict_data*, *result_list*)

s3vaultlib.editor.utils.**json_fixer**(*json_data*)

s3vaultlib.editor.utils.**yaml_fixer**(*yaml_data*)

s3vaultlib.editor.validators module

class s3vaultlib.editor.validators.**JSONValidator**
Bases: prompt_toolkit.validation.Validator

static validate(*document*)

class s3vaultlib.editor.validators.**YAMLValidator**
Bases: prompt_toolkit.validation.Validator

validate(*document*)

Module contents

s3vaultlib.kms package

Submodules

s3vaultlib.kms.kmsresolver module

class s3vaultlib.kms.kmsresolver.**KMSResolver**(*connection_manager*, *keyalias=''*, *role_name=''*)

Bases: object

Object that resolves the KMS key associated to a role, or load a keyarn with a specified alias

retrieve_key_arn()

Return the KMS arn of a key

Returns key arn

Return type basestring

exception s3vaultlib.kms.kmsresolver.**KMSResolverException**

Bases: exceptions.Exception

Module contents

s3vaultlib.metadata package

Submodules

s3vaultlib.metadata.base module

class s3vaultlib.metadata.base.**MetadataBase**(*session_info=None*)

Bases: object

account_id

instance_id

region

role

s3vaultlib.metadata.ec2 module

class s3vaultlib.metadata.ec2.**EC2Metadata**(*endpoint='169.254.169.254'*, *version='latest'*, *session_info=None*)

Bases: *s3vaultlib.metadata.base.MetadataBase*

Object that retrieve metadata from within an EC2 instance

account_id

Return the account_id associated to the instance

Returns account_id

Return type basestring

instance_id

Return the instance_id associated to the instance

Returns instance_id

Return type basestring

region
Return the region associated to the instance

Returns region

Return type basestring

role
Return the role associated to the instance

exception s3vaultlib.metadata.ec2.**EC2MetadataException**
Bases: exceptions.Exception

s3vaultlib.metadata.factory module

class s3vaultlib.metadata.factory.**MetadataFactory**
Bases: object

static get_instance(*is_ec2=False*, *session_info=None*)

s3vaultlib.metadata.local module

class s3vaultlib.metadata.local.**LocalMetadata**(*session_info=None*)
Bases: *s3vaultlib.metadata.base.MetadataBase*

account_id

instance_id

region

role

exception s3vaultlib.metadata.local.**LocalMetadataException**
Bases: exceptions.Exception

Module contents

s3vaultlib.s3 package

Submodules

s3vaultlib.s3.s3fs module

class s3vaultlib.s3.s3fs.**S3Fs**(*connection_factory*, *bucket*, *path=''*)
Bases: object

Object that abstracts operation with encrypted objects on S3

fs = None

Type pyboto3.s3

get_object(*name*)
Return a s3fsobject identified by name

Parameters **name** – object name

Returns s3fsobject

Return type *S3FsObject*

static is_file(*s3elem*)
Return true is an s3 json element represents a valid file

objects
Return a list of s3fsobjects

put_object(*name*, *content*, *encryption_key_arn*, *force_dot_file=False*)
Put an object in the S3 path by encrypting it with SSE

Parameters

- **name** – object name
- **content** – content of the object
- **encryption_key_arn** – key arn to use for encryption
- **force_dot_file** – if enabled it disable the check with dot in the file

Returns the created s3object

Return type *S3FsObject*

update_s3fsobject(*s3fsobject*)
Update an S3FSObject

Parameters **s3fsobject** – S3FsObject to update

Type *S3FsObject*

Returns the updated object

Return type *S3FsObject*

exception s3vaultlib.s3.s3fs.**S3FsException**
Bases: exceptions.Exception

exception s3vaultlib.s3.s3fs.**S3FsObjectNotFoundException**
Bases: *s3vaultlib.s3.s3fs.S3FsException*

s3vaultlib.s3.s3fsobject module

class s3vaultlib.s3.s3fsobject.**S3FsObject**(*data*, *bucket*, *path*, *fs*)
Bases: object

Implement the S3FsObject, an abstraction around a S3 file with SSE encryption

is_encrypted
Return true if the object is encrypted

Returns True or False

Return type bool

static is_json(*data*)
Return True if the content is a valid json

Parameters **data** – content to evaluate

Returns True or False

Return type bool

kms_arn
Return the KMS ARN used to encrypt the object

Returns KMS ARN

Return type basestring

metadata
Return the metadata associated with the object (file metadata)

Returns medatata

Return type dict

raw()

exception s3vaultlib.s3.s3fsobject.**S3FsObjectException**
Bases: exceptions.Exception

Module contents

s3vaultlib.template package

Submodules

s3vaultlib.template.templatefile module

class s3vaultlib.template.templatefile.**TemplateFile**(*filename*)
Bases: object

filename

get_raw_copy_src()

is_raw_copy(*s3fs_objects*)
Detect if the template represent a raw copy of the file

Parameters

- **template_file** –
- **s3fs_objects** –

Returns

template_data

exception s3vaultlib.template.templatefile.**TemplateFileException**
Bases: exceptions.Exception

s3vaultlib.template.templaterenderer module

class s3vaultlib.template.templaterenderer.**TemplateRenderer**(*template_file*, *s3fs*)
Bases: object

Renders a template based on S3Fs location

render(**kwargs*)
Renders the template

Parameters **kwargs** – additional variables to use in the rendering

Returns content of the rendered template

Return type basestring

Module contents

s3vaultlib.utils package

Submodules

s3vaultlib.utils.io module

s3vaultlib.utils.io.**write_with_modecheck**(*file_handler*, *data*)

s3vaultlib.utils.yaml module

s3vaultlib.utils.yaml.**build_parser**(**args*, ***kwargs*)

s3vaultlib.utils.yaml.**load_from_stream**(*stream*)

s3vaultlib.utils.yaml.**write_to_string**(*data*)

s3vaultlib.utils.yaml.**write_to_file**(*data*, *filename*)

exception s3vaultlib.utils.yaml.**ParserError**(*context=None*, *context_mark=None*, *problem=None*, *problem_mark=None*, *note=None*, *warn=None*)

Bases: ruamel.yaml.error.MarkedYAMLError

exception s3vaultlib.utils.yaml.**ScannerError**(*context=None*, *context_mark=None*, *problem=None*, *problem_mark=None*, *note=None*, *warn=None*)

Bases: ruamel.yaml.error.MarkedYAMLError

Module contents

7.1.2 Submodules

7.1.3 s3vaultlib.cli module

s3vaultlib.cli.**check_args**()

Check the args from the command line

Returns args object

s3vaultlib.cli.**configure_logging**(*level*)

Configure the logging level of the tool

Parameters **level** – level to set

Returns

s3vaultlib.cli.**main**()

Command line tool to use some functionality of the S3Vault

Returns

s3vaultlib.cli.**validate_args**(*parser*)

Returns

7.1.4 s3vaultlib.commands module

s3vaultlib.commands.**is_ec2**(*args*)

s3vaultlib.commands.**command_ansiblepath**()

s3vaultlib.commands.**command_configedit**(*args*, *conn_manager*)

s3vaultlib.commands.**command_configset**(*args*, *conn_manager*)

s3vaultlib.commands.**command_createcloudformation**(*args*)

s3vaultlib.commands.**command_createconfig**(*args*)

s3vaultlib.commands.**command_createtoken**(*args*, *conn_manager*)

s3vaultlib.commands.**command_get**(*args*, *conn_manager*)

s3vaultlib.commands.**command_push**(*args*, *conn_manager*)

s3vaultlib.commands.**command_template**(*args*, *conn_manager*)

7.1.5 s3vaultlib.s3vaultlib module

class s3vaultlib.s3vaultlib.**S3Vault**(*bucket*, *path*, *connection_factory=None*, *is_ec2=False*)

Bases: object

Implements a Vault by using S3 as backend and KMS as way to protect the data

create_config_property(*configfile*, *encryption_key_arn=''*, *key_alias=''*, *role_name=''*)

Create a configuration file in the S3Vault

Parameters

- **configfile** – configuration file name
- **encryption_key_arn** – KMS Arn to use
- **key_alias** – KMS Alias to use
- **role_name** – Role to use to resolve the KMS Key

Returns s3fsobject

Return type *S3FsObject*

get_file(*name*)

Get a file from S3Vault

Parameters **name** – filename

Returns file content

Return type basestring

get_file_metadata(*name*)

Get a file from S3Vault

Parameters **name** – filename

Returns file content

Return type dict

get_property(*configfile*, *key*)
Get a configuration property from a config file from the S3Vault

Parameters

- **configfile** – configuration file
- **key** – key to query

Returns value of the key

put_file(*src*, *dest*, *encryption_key_arn=''*, *key_alias=''*, *role_name=''*)
Upload a file to the S3Vault

Parameters

- **src** – source file name
- **dest** – destination file name
- **encryption_key_arn** – KMS Key arn to use
- **key_alias** – KMS Key alias to use
- **role_name** – Role from which resolve the key

Returns metadata of the uploaded object

Return type dict

render_template(*template_file*, ***kwargs*)
Renders a template file using the information available in the S3Vault

Parameters

- **template_file** – file name to use as template
- **kwargs** – additional variables to use in the rendering

Returns rendered content

Return type basestring

set_property(*configfile*, *key*, *value*, *encryption_key_arn=''*, *key_alias=''*, *role_name=''*)
Set a property in a configuration file in the S3Vault

Parameters

- **configfile** – configfile name
- **key** – key
- **value** – value
- **encryption_key_arn** – KMS Key to use
- **key_alias** – KMS alias to use
- **role_name** – Role to use to resolve the KMS Key

Returns metadata of the config file created/updated

Return type basestring

exception s3vaultlib.s3vaultlib.**S3VaultException**
Bases: exceptions.Exception

exception s3vaultlib.s3vaultlib.**S3VaultObjectNotFoundException**
Bases: exceptions.Exception

7.1.6 Module contents

Top-level package for S3Vault Library.

CHAPTER 8

Contributing

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

You can contribute in many ways:

8.1 Types of Contributions

8.1.1 Report Bugs

Report bugs at https://github.com/s3vaultlib/s3vaultlib/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

8.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

8.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

8.1.4 Write Documentation

S3Vault Library could always use more documentation, whether as part of the official S3Vault Library docs, in docstrings, or even on the web in blog posts, articles, and such.

8.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/s3vaultlib/s3vaultlib/issues.

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.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

8.2 Get Started!

Ready to contribute? Here's how to set up *s3vaultlib* for local development.

1. Fork the *s3vaultlib* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/s3vaultlib.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv s3vaultlib
$ cd s3vaultlib/
$ python setup.py develop
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 s3vaultlib tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

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

7. Submit a pull request through the GitHub website.

8.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/s3vaultlib/s3vaultlib/pull_requests and make sure that the tests pass for all supported Python versions.

8.4 Tips

To run a subset of tests:

```
$ py.test tests.test_s3vaultlib
```

CHAPTER 9

Credits

9.1 Development Lead

- Giuseppe Chiesa <mail@giuseppechiesa.it>

9.2 Contributors

None yet. Why not be the first?

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

S

T

U

V

W

Y