pendant

Release 0.4.1

Dec 10, 2018

Contents

```
pip install pendant
```

CHAPTER 1

Features

- Submit Batch jobs

1.1 pendant package

1.1.1 `aws` Submodule

pendant.aws module

`pendant.aws.`**`cli`**(*command: str*) → str

Use the `awscli` to execute a command.

This function will call the `awscli` within the same process and not spawn subprocesses. In addition, the STDERR of the called function will be surpressed and the STDOUT will be returned as a string.

Parameters **command** – The command to be executed by the `awscli`.

Examples

```
>>> # cli('--version')
```

pendant.aws.batch module

`class` `pendant.aws.batch.`**`JobDefinition`**

Bases: `object`

A Batch job definition.

`name`

Return the name of the job definition.

parameters
Return the parameters of the job definition.

revision
Return the revision of the job definition.

validate() → None
Validate this job definition after initialization.

at_revision(*revision: str*) → pendant.aws.batch.JobDefinition
Set this job definition to a specific revision.

make_job_name(*moment: Optional[datetime.datetime] = None*) → str
Format a Batch job name from this definition.

to_dict() → Dict[str, str]
Return a dictionary of all parameters and their values as strings.

class pendant.aws.batch.**BatchJob**(*definition: pendant.aws.batch.JobDefinition*)
Bases: object

An AWS Batch job.

A Batch job can be instantiated and then submitted against the Batch service. After submission, the job's status can be queried, the job's logs can be read, and other methods can be called to understand the state of the job.

Parameters **definition** – A Batch job definition.

container_overrides
Return container overriding parameters.

job_id
Return the job ID.

queue
Return the job queue.

static describe_job(*job_id: str*) → Dict
Describe this job.

static describe_jobs(*job_ids: List[str]*) → List[Dict]
Describe a Batch job by job ID.

status() → str
Return the job status.

cancel(*reason: str*) → Dict
Cancel this job.

Parameters **reason** – The reason why the job must be canceled.

Returns The service response to job cancellation.

terminate(*reason: str*) → Dict
Terminate this job.

Jobs that are in the STARTING or RUNNING state are terminated, which causes them to transition to FAILED. Jobs that have not progressed to the STARTING state are cancelled.

Parameters **reason** – The reason why the job must be terminated.

Returns The service response to job termination.

is_running() → bool
Return if this job's state is RUNNING or not.

is_runnable() → bool
Return if this job's state is RUNNABLE or not.

is_submitted() → bool
Return if this job has been submitted to Batch.

submit(*queue: str*, *container_overrides: Optional[Mapping] = None*) → pendant.aws.response.SubmitJobResponse
Submit this job to Batch.

Parameters

- **queue** – The Batch job queue to use.
- **container_overrides** – The values to override in the spawned container.

Returns The service response to job submission.

log_stream_name() → str
Return the Batch log stream name for this job.

log_stream_events() → List[pendant.aws.logs.LogEvent]
Return all log events for this job.

Returns *events* – All log events, to date.

pendant.aws.exception module

exception pendant.aws.exception.**BatchJobNotFoundError**
Bases: Exception

A Batch job not found error.

exception pendant.aws.exception.**BatchJobSubmissionError**
Bases: Exception

A Batch job submission error.

exception pendant.aws.exception.**LogStreamNotFoundError**
Bases: Exception

A log stream not found error.

exception pendant.aws.exception.**S3ObjectNotFoundError**
Bases: Exception

A file not found error for objects on S3.

pendant.aws.logs module

class pendant.aws.logs.**LogEvent**(*record: Mapping*)
Bases: object

A AWS Cloudwatch log event.

Parameters **record** – A dictionary of log metadata.

class pendant.aws.logs.**AwsLogUtil**
Bases: object

AWS Cloudwatch cloud utility functions.

get_log_events(*group_name: str*, *stream_name: str*) → List[pendant.aws.logs.LogEvent]
Get all log events from a stream within a group.

pendant.aws.response module

class pendant.aws.response.**AwsResponse**
Bases: `object`

A generic HTTP response from AWS.

class pendant.aws.response.**SubmitJobResponse**(*response: Mapping*)
Bases: *pendant.aws.response.AwsResponse*

A Batch submit-job response.

is_ok() → bool
Return if response was successful.

http_code() → int
Return the HTTP status code of this response.

pendant.aws.s3 module

class pendant.aws.s3.**S3Uri**(*path: Union[str, S3Uri]*)
Bases: `object`

An S3 URI which conforms to RFC 3986 formatting.

Parameters **path** – The S3 URI path.

Examples

```
>>> uri = S3Uri('s3://mybucket/prefix')
>>> uri.scheme
's3://'
>>> uri.bucket
'mybucket'
>>> uri / 'myobject'
S3Uri('s3://mybucket/prefix/myobject')
```

delimiter = '/'

scheme
Return the RFC 3986 scheme of this URI.

Example

```
>>> uri = S3Uri('s3://mybucket/myobject')
>>> uri.scheme
's3://'
```

bucket
Return the S3 bucket of this URI.

Example

```
>>> uri = S3Uri('s3://mybucket/myobject')
>>> uri.bucket
'mybucket'
```

key
Return the S3 key of this URI.

Example

```
>>> uri = S3Uri('s3://mybucket/myobject')
>>> uri.key
'myobject'
```

add_suffix(*suffix: str*) → pendant.aws.s3.S3Uri
Add a suffix to this S3 URI.

Parameters **suffix** – Append this suffix to the URI.

Examples

```
>>> uri = S3Uri('s3://mybucket/myobject.bam')
>>> uri.add_suffix('.bai')
S3Uri('s3://mybucket/myobject.bam.bai')
```

This is equivalent to:

```
>>> S3Uri('s3://mybucket/myobject.bam') + '.bai'
S3Uri('s3://mybucket/myobject.bam.bai')
```

object_exists() → bool
Test if this URI references an object that exists.

pendant.aws.s3.**s3api_head_object**(*bucket: str*, *key: str*, *profile: str = 'default'*) → Dict
Use the `awscli` to make a GET request on an S3 object's metadata.

Parameters

- **bucket** – The S3 bucket name.
- **key** – The S3 object key.
- **profile** – The AWS profile to use, defaults to *"default"*.

Returns A dictionary of object metadata, if the object exists.

pendant.aws.s3.**s3api_object_exists**(*bucket: str*, *key: str*, *profile: str = 'default'*) → bool
Use the `awscli` to test if an S3 object exists.

Parameters

- **bucket** – The S3 bucket name.
- **key** – The S3 object key.
- **profile** – The AWS profile to use, defaults to *"default"*.

`pendant.aws.s3.`**`s3_object_exists`**`(`*bucket: str*, *key: str*`)` → bool

Use `boto3.S3.Object` to test if an S3 object exists.

Parameters

- **bucket** – The S3 bucket name.
- **key** – The S3 object key.

1.1.2 `util` Submodule

pendant.util module

`class` `pendant.util.`**`ExitCode`**

Bases: `int`

The code returned to a parent process by an executable.

Examples

```
>>> from subprocess import call
>>> exit_code = ExitCode(call('ls'))
>>> exit_code.is_ok()
True
```

`is_ok``()` → bool

Is this code zero.

`pendant.util.`**`format_ISO8601`**`(`*moment: datetime.datetime*`)` → str

Format a datetime into a filename compatible IS8601 representation.

Parameters **moment** – A datetime.

Returns The ISO8601 datetime formatted with hyphens as seperators.

Examples

```
>>> from datetime import datetime
>>> format_ISO8601(datetime(2018, 2, 23, 12, 13, 38))
'2018-02-23T12-13-38'
```

1.2 How to Contribute

Pull requests, feature requests, and issues welcome! The complete test suite is configured through `Tox`:

```
cd pendant
pip install tox
tox  # Run entire dynamic / static analysis test suite
```

List all environments with:

```
 tox -av
using tox.ini: .../pendant/tox.ini
using tox-3.1.2 from ../tox/__init__.py
default environments:
py36      -> run the test suite with (basepython)
py36-lint -> check the code style
py36-type -> type check the library
py36-docs -> test building of HTML docs

additional environments:
dev       -> the official sample_sheet development environment
```

To run just one environment:

```
tox -e py36
```

To pass in positional arguments to a specified environment:

```
tox -e py36 -- -x tests/test_sample_sheet.py
```

Python Module Index

Index