Bitbucket Pipes Toolkit

Jun 27, 2023

Contents:

This library provides various utilites for writing and testing Bitbucket pipes.

CHAPTER 1

Indices and tables

1.1 Installation

```
pip install bitbucket_pipes_toolkit
```

1.2 Available classes and funcions

bitbucket_pipes_toolkit.Pipe([...])	Base class for all pipes.

bitbucket_pipes_toolkit.**configure_logger**()
Configure logger to produce colorized output.

bitbucket_pipes_toolkit.**get_variable**(*name*, *required=False*, *default=None*)
Fetch the value of a pipe variable.

Parameters

- **name** (*str*) – Variable name.
- **required** (*bool, optional*) – Throw an exception if the env var is unset.
- **default** (str, optional) – Default value if the env var is unset.

Returns Value of the variable

Raises Exception: If a required variable is missing.

bitbucket_pipes_toolkit.**required**(*name*)
Get the value of a required pipe variable.

This function is basically an alias to get_variable with the required parameter set to True.

Parameters **name** (*str*) – Variable name.

Returns Value of the variable

Raises Exception: If a required variable is missing.

bitbucket_pipes_toolkit.**enable_debug**()
Enable the debug log level.

bitbucket_pipes_toolkit.**success**(*message='Success'*, *do_exit=True*)
Prints the colorized success message (in green)

Parameters

- **message** (*str, optional*) – Output message
- **do_exit** (*bool, optional*) – Call sys.exit if set to True

bitbucket_pipes_toolkit.**fail**(*message='Fail!'*, *do_exit=True*)
Prints the colorized failure message (in red)

Parameters

- **message** (*str, optional*) – Output message
- **do_exit** (*bool, optional*) – Call sys.exit if set to True

class bitbucket_pipes_toolkit.**Pipe**(*pipe_metadata=None*, *pipe_metadata_file=None*, *schema=None*, *env=None*, *logger=<RootLogger root (INFO)>*, *check_for_newer_version=False*)
Base class for all pipes. Provides utilities to work with configuration, validation etc.

variables
Dictionary containing the pipes variables.

Type dict

schema
Dictionary with the pipe parameters schema in the cerberus format.

Type dict

env
Dict-like object containing pipe parameters. This is usually the environment variables that you get from os.environ

Type dict

Parameters

- **pipe_metadata** (*dict*) – Dictionary containing the pipe metadata
- **pipe_metadata_file** (*str*) – Path to .yml file with metadata
- **schema** (*dict*) – Schema for pipe variables.

Pipe variables validation Pip variables validation is done at the time of initializing a pipe. The environment variables are validated against the schema provided. See https://docs.python-cerberus.org/en/stable/ for more details on how to specify schemas.

check_for_newer_version()
Check if a newer version is available and show a warning message

```
>>> metadata = {'image': 'bitbucketpipelines/aws-ecs-deploy:0.0.3',
...              'repository': 'https://bitbucket.org/atlassian/aws-ecs-deploy
↪'}
>>> pipe = Pipe(pipe_metadata=metadata, schema={})
>>> pipe.check_for_newer_version()
True
```

enable_debug_log_level()
Enable the DEBUG log level.

fail(*message*, *print_community_link=False*)
Fail the pipe and exit.

Parameters

- **message** (*str*) – Error message to show.
- **print_community_link** (*bool*) – print or not.

get_community_link()
Retrieve link to create a community question with predefined tags.

get_pipe_name()
Retrive a pipe's name from pipe's repository url.

```
>>> metadata = {'image': 'bitbucketpipelines/aws-ecs-deploy:0.0.3',
...              'repository': 'https://bitbucket.org/atlassian/aws-ecs-deploy
↪'}
>>> pipe = Pipe(pipe_metadata=metadata, schema={})
>>> pipe.get_pipe_name()
'atlassian/aws-ecs-deploy'
```

get_variable(*name*)
Retrieve a pipe variable.

Parameters **name** (*str*) – The name of a variable.

Returns The value of the variable.

log_debug(*message*)
Log a debug message

```
>>> pipe = Pipe(schema={})
>>> pipe.log_debug('hello')
```

log_error(*message*)
Log an error message

```
>>> pipe = Pipe(schema={})
>>> pipe.log_error('hello')
```

log_info(*message*)
Log an info message

```
>>> pipe = Pipe(schema={})
>>> pipe.log_info('hello')
```

log_warning(*message*)
Log a warning message

```
>>> pipe = Pipe(schema={})
>>> pipe.log_warning('hello')
```

resolve_auth()
Resolve authorization. Currently supported: - BITBUCKET_USERNAME and BITBUCKET_APP_PASSWORD or - BITBUCKET_ACCESS_TOKEN

Returns <HTTPBasicAuth object> or <TokenAuth object>

Raises Authentication missing ... SystemExit: 1

```
>>> pipe = Pipe(schema={})
>>> pipe.resolve_auth()
Traceback (most recent call last):
SystemExit: 1
```

```
>>> pipe = Pipe(schema={})
>>> pipe.variables = {'BITBUCKET_USERNAME': 'test', 'BITBUCKET_APP_PASSWORD':
↪'test-pass'}
>>> pipe.resolve_auth() # doctest: +ELLIPSIS
<...HTTPBasicAuth object at 0x...>
```

```
>>> pipe = Pipe(schema={})
>>> pipe.variables = {'BITBUCKET_ACCESS_TOKEN': 'test-token'}
>>> pipe.resolve_auth() # doctest: +ELLIPSIS
<...TokenAuth object at 0x...>
```

run()
Run the pipe.

The main entry point for a pipe execution. This will do all the setup steps, like enabling debug mode if configure etc.

static success(*message*, *do_exit=False*)
Show a success message.

Parameters

- **message** (*str*) – Message to print
- **do_exit** (*bool*) – Call sys.exit or not

validate()
Validates the environment variables against a provided schema.

Variable schema is a dictionary in a cerberus format. See https://docs.python-cerberus.org/en/stable/ for more details about this library and validation rules.

class bitbucket_pipes_toolkit.**ArrayVariable**(*name*, *env*)
Example:

```
>>> env = {
...         'EXTRA_ARGS_COUNT': 3,
```

(continues on next page)

(continued from previous page)

```
...         'EXTRA_ARGS_0': '--description',
...         'EXTRA_ARGS_1': 'text containing spaces',
...         'EXTRA_ARGS_2': '--verbose'}
>>>
>>> array_variable = ArrayVariable('EXTRA_ARGS', env)
>>> array_variable[2]
'--verbose'
```

append(*value*)
S.append(value) – append value to the end of the sequence

clear() → None – remove all items from S

count(*value*) → integer – return number of occurrences of value

decompile()
Example

```
>>> var = ArrayVariable.from_list('TEST', ['foo','bar', 'baz'])
>>> var.decompile()
{'TEST_0': 'foo', 'TEST_1': 'bar', 'TEST_2': 'baz', 'TEST_COUNT': 3}
```

extend(*values*)
S.extend(iterable) – extend sequence by appending elements from the iterable

index(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

insert(*index*, *value*)
S.insert(index, value) – insert value before index

pop([*index*]) → item – remove and return item at index (default last).
Raise IndexError if list is empty or index is out of range.

remove(*value*)
S.remove(value) – remove first occurrence of value. Raise ValueError if the value is not present.

reverse()
S.reverse() – reverse *IN PLACE*

class bitbucket_pipes_toolkit.**CodeInsights**(*repo*, *username*, *auth_type='authless'*, *auth_proxy_host='host.docker.internal'*, *app_password=None*, *account=None*)
This is a wrapper class that uses Bitbucket CodeInsights API to create Code reports.

Parameters

- **auth_type** (*str*) – The type of authentication mechanism (basic, authless)
- **auth_proxy_host** (*str*) – Hostname of the pipelines authentication proxy.
- **account** (*str*) – Account, that will be used to authenticate the API requests.
- **app_password** (*str*) – Bitbucket app password created for a user who owns the account.
- **username** (*str*) – The name of the account that owns the repository.
- **repo** (*str*) – Repository slug.
- **the authless authentication type the pipe will proxy API requests via internal authentication** (*With*) –

proxy. In that case username and password are not required. To create an annotation without authentication in pipelines, pass the 'localhost' as the auth_proxy_host

parameter.

create_annotation(*commit*, *report_id*, *annotation_data*)
Send a request to create an annotation for a given commit and report id.

Parameters

- **commit** (*str*) – Hash of the commit to create annotation for.
- **report_id** (*str*) – Report id.
- **annotation_data** (*dict*) – Dictionary with the data to create an annotation.

Example of annotation data: annotation_data = {

"type": "report_annotation", "annotation_type": "BUG", "uuid": "{asdasd-1231321-asdad-2131321}", "external_id": "100", # required "summary": "This line has a bug!", # required "details": "This is a really bad bug, caused by IEEE09999, you need to ensure arrays dont go out of bounds.", "path": "src/main/java/com/atlassian/pipelines/Service.java", "line": 10, "severity": "HIGH", "result": "FAILED", "link": "https://bug-tool.atlassian.com/report/10/bug/100", "created_on": "2020-01-08T00:56:20.593Z", "updated_on": "2020-01-09T12:00:10.123Z"

}

Returns Response json data.

Return type json

create_report(*commit*, *report_data*)
Send a request to create a report for a given commit.

Parameters

- **commit** (*str*) – Hash of the commit to create report for.
- **report_data** (*dict*) – Dictionary with the data to create a report.

Example of report data: report_data = {

"type": "report", "uuid": "{asdasd-565656-asdad-565655}", "report_type": "BUG", "external_id": "10", # required "title": "Bug report", # required "details": "This bug report is auto generated by bug tool.", # required "result": "FAILED", "reporter": "Created by atlassians bug tool.", "link": "https://bug-tool.atlassian.com/report/10", "logo_url": "https://bug-tool.atlassian.com/logo.png", "data": [

{ "title": "FAILED", "type": "BOOLEAN", "value": true },

], "created_on": "2020-01-08T00:56:20.593Z", "updated_on": "2020-01-09T12:00:10.123Z"

}

Returns Response json data.

Return type json

delete_report(*commit*, *report_id*)
Send a request to delete report for a given commit.

Parameters

- **commit** (*str*) – Hash of the commit to delete report for.
- **report_id** (*str*) – Report id.

Returns True if response is ok.

Return type bool

get_report(*commit*, *report_id*)
Send a request to get report for a given commit.

Parameters

- **commit** (*str*) – Hash of the commit to get report for.
- **report_id** (*str*) – Report id.

Returns Response json data.

Return type json

get_reports(*commit*)
Send a request to get all the reports for a given commit.

Parameters **commit** (*str*) – Hash of the commit to get reports for.

Returns Response json data.

Return type json

1.3 Examples

Example subclissing and running the pipe:

```
from bitbucket_pipes_toolkit import Pipe

class MyPipe(Pipe):
    def run(self):
        super().run()
        print("I'm a simple little pipe")


# below is a simple schema for pipe variables.
schema = {'USERNAME': {'required': True, 'type': 'string'},
          'PASSWORD': {'required': True, 'type': 'string'}}

my_pipe=MyPipe(pipe_metadata='pipe.yml', schema=schema)
my_pipe=pipe.run()
```

Example writing a simple test case for a pipe

```
from bitbucket_pipes_toolkit.test import PipeTestCase

class PipelineTypeTestCase(TriggerBuildBaseTestCase):

def test_something_is_successful(self):
    result = self.run_container(environment={
        'USERNAME': 'JohnDoe',
        'PASSWORD': os.getenv('PASSWORD'),
    })

    self.assertRegexpMatches(
        result, r"Something was successful!")
```

Python Module Index

b

Index

R

S

V