
pytest-kafka Documentation

Release 0.4.0

Karoline Pauls

Aug 24, 2021

CONTENTS:

1	pytest-kafka	1
1.1	System requirements	2
1.2	Development	2
1.3	Acknowledgements	2
2	pytest_kafka	3
2.1	pytest_kafka package	3
3	Indices and tables	5
	Python Module Index	7
	Index	9

PYTEST-KAFKA

Pytest fixture factories for Zookeeper, Kafka server and Kafka consumer. [Read the API docs.](#)

```
from pathlib import Path
from pytest_kafka import (
    make_zookeeper_process, make_kafka_server, make_kafka_consumer,
    terminate,
)

ROOT = Path(__file__).parent
KAFKA_SCRIPTS = ROOT / 'kafka/bin/'
KAFKA_BIN = str(KAFKA_SCRIPTS / 'kafka-server-start.sh')
ZOOKEEPER_BIN = str(KAFKA_SCRIPTS / 'zookeeper-server-start.sh')

# You can pass a custom teardown function (or parametrise ours). Just don't call it_
↪ `teardown`
# or Pytest will interpret it as a module-scoped teardown function.
teardown_fn = partial(terminate, signal_fn=Popen.kill)
zookeeper_proc = make_zookeeper_process(ZOOKEEPER_BIN, teardown_fn=teardown_fn)
kafka_server = make_kafka_server(KAFKA_BIN, 'zookeeper_proc', teardown_fn=teardown_fn)
kafka_consumer = make_kafka_consumer(
    'kafka_server', seek_to_beginning=True, kafka_topics=['topic'])
```

This creates 3 fixtures:

1. `zookeeper_proc` - Zookeeper process
2. `kafka_server` - Kafka process
3. `kafka_consumer` - usable `kafka.KafkaConsumer` instance

`ZOOKEEPER_BIN` and `KAFKA_BIN` are paths to launch scripts in your Kafka distribution. Check this project's `setup.py` to see a way of installing Kafka for development.

It is advised to pass `seek_to_beginning=True` because otherwise some messages may not be captured by the consumer. This requires knowing the topics upfront because without topics there's no partitions to seek.

Kafka server is known to take a couple of seconds to terminate gracefully. You probably don't need that, so you can pass `partial(terminate, signal_fn=Popen.kill)` to make it killed with `SIGKILL` and waited for afterwards.

It's possible to create multiple Kafka fixtures forming a cluster by passing the same Zookeeper fixture to them. For an example, check the [tests](#).

Session-scoped fixtures are also available. Consult the [test suite](#).

1.1 System requirements

- Python 3.6+
- a JVM that can run Kafka and Zookeeper
- Kafka - <https://kafka.apache.org/downloads>

1.2 Development

Execute `./setup.py develop` in a virtualenv. This will take care of:

- installing dependencies
- updating pip
- installing dev dependencies
- installing Kafka to the project dir (for development only)

1.3 Acknowledgements

The library has been open-sourced from a codebase belonging to [Infectious Media](#).

PYTEST_KAFKA

2.1 pytest_kafka package

Pytest-kafka public API.

```
pytest_kafka.make_zookeeper_process(zk_bin, zk_port=None,
                                     zk_config_template='dataDir={zk_data_dir}\nclientPort={zk_port}\nmaxClientC
                                     teardown_fn=<function terminate>, scope='function')
```

Make a Zookeeper fixture.

The fixture will spawn a Zookeeper process in a new process group and return its process handle and port number. Data will be stored in a Pytest-provided temporary directory.

Parameters

- **zk_bin** (`str`) – path to Zookeeper launch script (typically to bin/zookeeper-server-start.sh)
- **zk_port** (`Optional[int]`) – Zookeeper port (random free port by default)
- **zk_config_template** (`str`) – Zookeeper config template, must use keys `zk_data_dir` and `zk_port`. See `pytest_kafka.constants.ZOOKEEPER_CONFIG_TEMPLATE`.
- **teardown_fn** (`Callable[[Popen], Any]`) – function to tear down Zookeeper (`terminate()` by default)
- **scope** (`str`) – ‘function’ or ‘session’

Return type `Callable[..., Tuple[Popen, int]]`

```
pytest_kafka.make_kafka_server(kafka_bin, zookeeper_fixture_name, kafka_port=None,
                               kafka_config_template='reserved.broker.max.id=65535\nbroker.id={kafka_port}\nlisteners=1\nlog.dirs={kafka_log_dir}\nnum.network.threads=1\nnum.io.threads=1\nnum.recovery.threads.per.data.dir=1\nzookeeper.connect={zk_host}:{zk_port}\nzookeeper.session.timeout=15\nzookeeper.sync.time.out=15')
```

Make a Kafka fixture.

The fixture will spawn a Kafka process in a new process group and return its process handle and port number. Data will be stored in a Pytest-provided temporary directory.

Parameters

- **zookeeper_fixture_name** (`str`) – the name of the Zookeeper fixture to depend on. The scope must not be wider than this fixture’s scope.
- **kafka_bin** (`str`) – path to Kafka launch script (typically to bin/kafka-server-start.sh)

- **kafka_port** (`Optional[int]`) – Kafka port (random free port by default)
- **kafka_config_template** (`str`) – Kafka config template, must use keys `kafka_log_dir` and `kafka_port`. See `pytest_kafka.constants.KAFKA_SERVER_CONFIG_TEMPLATE`.
- **teardown_fn** (`Callable[[Popen], Any]`) – function to tear down Kafka (`terminate()` by default)
- **scope** (`str`) – ‘function’ or ‘session’
- **timeout** (`int`) – How long to wait for kafka to come online in seconds

Return type `Callable[..., Tuple[Popen, int]]`

```
pytest_kafka.make_kafka_consumer(kafka_fixture_name, kafka_topics=None,
                                seek_to_beginning=False, scope='function', **consumer_kwargs)
```

Make a Kafka consumer fixture.

Parameters

- **kafka_fixture_name** (`str`) – the name of the Kafka fixture to depend on
- **kafka_topics** (`Optional[List[str]]`) – topics to subscribe to
- **seek_to_beginning** (`bool`) – whether the consumer should consume from the earliest offsets. Solves the race condition between consumer setup and Kafka server + Producer setup but requires to know the topics upfront.
- **consumer_kwargs** – what to pass to `KafkaConsumer`. By default `bootstrap_servers` will get the server from the passed fixture and `consumer_timeout_ms` will be `pytest_kafka.constants.DEFAULT_CONSUMER_TIMEOUT_MS`.

It’s recommended to pass both `kafka_topics` and `seek_to_beginning`.

Return type `Callable[..., KafkaConsumer]`

```
pytest_kafka.terminate(proc, signal_fn=<function Popen.terminate>, wait_timeout=5)
```

Kill the process with the desired method (SIGTERM by default) and wait for it.

Return type `None`

2.1.1 Submodules

pytest_kafka.constants module

Config templates and constants.

```
pytest_kafka.constants.DEFAULT_CONSUMER_TIMEOUT_MS = 500
```

Kafka Consumer timeout in milliseconds.

```
pytest_kafka.constants.KAFKA_SERVER_CONFIG_TEMPLATE = 'reserved.broker.max.id=65535\nbroker
```

Kafka config template. `kafka_log_dir`, `kafka_port`, and `zk_port` keys are required.

```
pytest_kafka.constants.ZOOKEEPER_CONFIG_TEMPLATE = 'dataDir={zk_data_dir}\nclientPort={zk_p
```

Zookeeper config template. `zk_data_dir` and `zk_port` keys are required.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`pytest_kafka`, 3

`pytest_kafka.constants`, 4

INDEX

D

`DEFAULT_CONSUMER_TIMEOUT_MS` (in module `pytest_kafka.constants`), 4

K

`KAFKA_SERVER_CONFIG_TEMPLATE` (in module `pytest_kafka.constants`), 4

M

`make_kafka_consumer()` (in module `pytest_kafka`), 4

`make_kafka_server()` (in module `pytest_kafka`), 3

`make_zookeeper_process()` (in module `pytest_kafka`), 3

P

`pytest_kafka` (module), 3

`pytest_kafka.constants` (module), 4

T

`terminate()` (in module `pytest_kafka`), 4

Z

`ZOOKEEPER_CONFIG_TEMPLATE` (in module `pytest_kafka.constants`), 4