

---

# **kser Documentation**

***Release 0.8.5***

**Cedric Dumay**

**Aug 13, 2018**



---

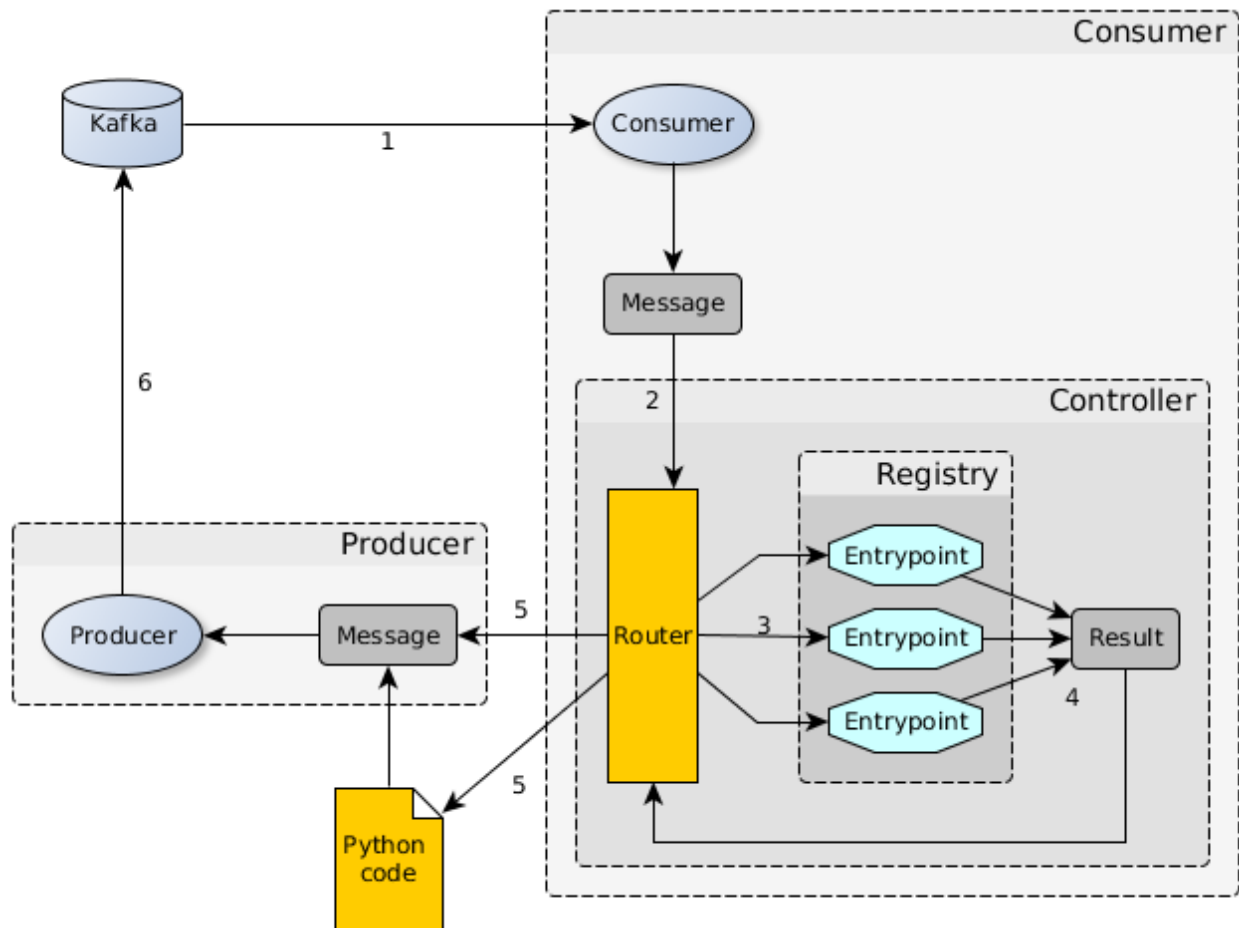
## Contents

---

<b>1</b>	<b>Features &amp; API Focus</b>	<b>3</b>
----------	---------------------------------	----------



Kser is a bundle of python library whose purpose is to serialize tasks to be executed on Kafka consumers.



1. A message comes from Kafka.
2. Consumer deserialize message and send it to the “router” witch dispatch the message to the registry.
3. Registry loads the correct entrypoint based on the message content.
4. Registry execute the entrypoint with the message data and return a result.
5. Result is sent back to the router which dispatch it.
6. Result may be sent back to kafka using the Producer.



## 1.1 kser.entry — Entries

**class** `kser.entry.Entrypoint` (*uuid=None, params=None, result=None*)

An entriypoint is the code which will be registred in the controller to handle execution.

**param str uuid** An unique identifier.

**param dict params** Entrypoint parameters

**param cdumay\_result.Result result** previous task result

**REQUIRED\_FIELDS**

Tuple or list of keys required by the entriypoint.

**check\_required\_params** ()

Perform a self test. It can be used to check params received, states... By default, this method check the presence of each item stored in `kser.entry.Entrypoint.REQUIRED_FIELDS` in the `kser.entry.Entrypoint.params` dictionary.

**execute** (*result=None*)

The main method used to launch the entriypoint execution. This method is execption safe. To execute an entriypoint without catching execption uses `kser.entry.Entrypoint.unsafe_execute()`.

**Parameters result** (*cdumay\_result.Result*) – Previous task result

**Returns** The execution result

**Return type** `cdumay_result.Result`

**classmethod from\_Message** (*kmsg*)

Initialize the entriypoint from a `kser.schemas.Message`

**Parameters kmsg** (*kser.schemas.Message*) – A message received from Kafka.

**Returns** The entriypoint

**Return type** `kser.entry.Entrypoint`

**log** (*message*, *level*=*logging.INFO*, \**args*, \*\**kwargs*)

Adds entrypoint information to the message and sends the result to `logging.log`.

**Parameters**

- **message** (*str*) – message content
- **level** (*int*) – [Logging Level](#)
- **args** (*list*) – Arguments which are merged into msg using the string formatting operator.
- **kwargs** (*dict*) – Keyword arguments.

**onerror** (*result*)

Trigger call on execution error.

**Parameters** **result** (*cdumay\_result.Result*) – Current execution result that led to the error.

**Returns** Return back the result

**Return type** *cdumay\_result.Result*

**onsuccess** (*result*)

Trigger call on execution success.

**Parameters** **result** (*cdumay\_result.Result*) – Current execution result that led to the success.

**Returns** Return back the result

**Return type** *cdumay\_result.Result*

**postinit** ()

Trigger call on execution post initialization.

**Parameters** **result** (*cdumay\_result.Result*) – Current execution result that led to the success.

**Returns** Return back the result

**Return type** *cdumay\_result.Result*

**postrun** (*result*)

Trigger call on execution post run. This trigger is called regardless of execution result.

**Parameters** **result** (*cdumay\_result.Result*) – Current execution result.

**Returns** Return back the result

**Return type** *cdumay\_result.Result*

**prerun** ()

Trigger call before the execution.

**run** ()

The entrypoint body intended to be overwrite.

**to\_Message** (*result*=*None*)

Serialize an entrypoint into a `kser.schemas.Message`.

**Parameters** **result** (*cdumay\_result.Result*) – Execution result.

**Returns** Return a message.

**Return type** `kser.schemas.Message`



**unsafe\_execute** (*result=None*)

Unlike `kser.entry.Entrypoint.execute()` this method launch the entrypoint execution without catching exception.

**Parameters** **result** (`cdumay_result.Result`) – Previous task result

**Returns** The execution result

**Return type** `cdumay_result.Result`

### 1.1.1 Example usage

Let's define a basic entrypoint:

```

1  import logging
2  from kser.entry import Entrypoint
3  from cdumay_result import Result
4
5  logging.basicConfig(
6      level=logging.DEBUG,
7      format="% (asctime)s % (levelname)-8s % (message)s"
8  )
9
10 class Hello(Entrypoint):
11     REQUIRED_FIELDS = ['name']
12
13     def run(self):
14         return Result(
15             uuid=self.uuid, stdout="Hello {name} !".format_map(self.params)
16         )

```

Execution result:

```

>>> Hello(params=dict(name="Cedric")).execute()
2018-02-21 18:26:46,762 DEBUG    Hello.PreRun: __main__.Hello[d455cba6-b329-4d2d-a4e5-
↳ 1fc2a0ff2781]
2018-02-21 18:26:46,762 DEBUG    Hello.Run: __main__.Hello[d455cba6-b329-4d2d-a4e5-
↳ 1fc2a0ff2781]
2018-02-21 18:26:46,762 DEBUG    Hello.PostRun: __main__.Hello[d455cba6-b329-4d2d-
↳ a4e5-1fc2a0ff2781]
2018-02-21 18:26:46,763 INFO     Hello.Success: __main__.Hello[d455cba6-b329-4d2d-
↳ a4e5-1fc2a0ff2781]: Hello Cedric !

```

Has we can see there is a required parameter *name*. Let's see what's happen if we didn't set it:

```

>>> Hello().execute()
2018-02-21 18:35:47,493 DEBUG    Hello.PreRun: __main__.Hello[f581fb61-0de1-489c-a0df-
↳ 2c03ce1d35b4]
2018-02-21 18:35:47,495 ERROR    Hello.Failed: __main__.Hello[f581fb61-0de1-489c-a0df-
↳ 2c03ce1d35b4]: Missing parameter: name

```

What's happen if we uses `kser.entry.Entrypoint.unsafe_execute()` instead of `kser.entry.Entrypoint.execute()`:

```

>>> Hello().unsafe_execute()
2018-02-21 18:39:23,522 DEBUG    Hello.PreRun: __main__.Hello[6aa38be5-cd82-441b-8853-
↳ 318545a053ad]

```

(continues on next page)

(continued from previous page)

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/kser/src/kser/entry.py", line 220, in unsafe_execute
    self._prerun()
  File "/opt/kser/src/kser/entry.py", line 147, in _prerun
    self.check_required_params()
  File "/opt/kser/src/kser/entry.py", line 54, in check_required_params
    raise ValidationError("Missing parameter: {}".format(param))
cdumay_rest_client.exceptions.ValidationError: Error 400: Missing parameter: name_
↪ (extra={})
```

See also:

**cdumay-result** A basic lib to serialize exception results.

**cdumay-rest-client** A basic REST client library.

## 1.2 kser.crypto — Message encryption

This module allow you to encrypt and decrypt messages in kafka

### 1.2.1 Install

```
pip install kser[crypto]
```

### 1.2.2 API focus

**kser.crypto.CryptoMessage(context) :**

It's a container which includes the original message as well as the nonce required by the consumer to decipher the content

**param dict context** We use marshmallow context to store the secretbox\_key

**kser.crypto.CryptoMessage.decode(jdata)**  
Encode message using libsodium

**Parameters kmsg** (*kser.schemas.Message*) – Kafka message

**Returns** the Encoded message

**kser.crypto.CryptoMessage.encode(kmsg)**  
Decode message using libsodium

**Parameters jdata** (*str*) – jdata to load

**Returns** the Encoded message

### 1.2.3 Example

For this example, we'll use [kafka-python](#) as kafka backend.

**Consumer example:**

```

1 from kser.controller import Controller
2 from kser.crypto import CryptoMessage
3 from kser.python_kafka.consumer import Consumer
4
5
6 class CryptoController(Controller):
7     TRANSPORT = CryptoMessage
8
9
10 class CryptoConsumer(Consumer):
11     REGISTRY = CryptoController
12
13
14 if __name__ == '__main__':
15     consumer = CryptoConsumer(config=dict(...), topics=list(...))
16     consumer.run()

```

### Producer example:

```

1 import os
2 from uuid import uuid4
3
4 from cdumay_result import Result
5 from kser.crypto import CryptoSchema
6 from kser.python_kafka.producer import Producer
7 from kser.schemas import Message
8
9
10 class CryptoProducer(Producer):
11     # noinspection PyUnusedLocal
12     def send(self, topic, kmsg, timeout=60):
13         result = Result(uuid=kmsg.uuid)
14         try:
15             self.client.send(topic, CryptoSchema(context=dict(
16                 secretbox_key=os.getenv("KSER_SECRETBOX_KEY", None)
17             )).encode(self._onmessage(kmsg)).encode("UTF-8"))
18
19             result.stdout = "Message {}[{}] sent in {}".format(
20                 kmsg.entrypoint, kmsg.uuid, topic
21             )
22             self.client.flush()
23
24         except Exception as exc:
25             result = Result.from_exception(exc, kmsg.uuid)
26
27         finally:
28             if result.retcode < 300:
29                 return self._onsuccess(kmsg=kmsg, result=result)
30             else:
31                 return self._onerror(kmsg=kmsg, result=result)
32
33
34 if __name__ == '__main__':
35     producer = CryptoProducer(config=dict(...))
36     producer.send("my.topic", Message(uuid=str(uuid4()), entrypoint="myTest"))

```

## 1.3 Operation and task

Kser provide a way to create and launch operations. An operation is a list of task executed linearly.

---

**Note:** Each operation's task may be executed on different consumer.

---

### 1.3.1 API focus

**class** `kser.sequencing.task.Task` (*uuid=None, params=None, status="PENDING", result=None, metadata=None*)

A task is a `kser.entry.EntryPoint` with additional attributes. To do this, use a database as shared backend.

**Parameters**

- **uuid** (*str*) – task unique identifier
- **params** (*dict*) – task parameter
- **status** (*str*) – task status
- **result** (*cdumay\_result.Result*) – forwarded result from a previous task
- **metadata** (*dict*) – task context

**log** (*message, level=logging.INFO, \*args, \*\*kwargs*)

Send log entry, prefixing message using the following format:

```
{TaskName}. {TaskStatus}: {EntryPointPath} [{TaskUUID}]:
```

**Parameters**

- **message** (*str*) – log message
- **level** (*int*) – [Logging level](#)
- **args** (*list*) – log record arguments
- **kwargs** (*dict*) – log record key arguments

**class** `kser.sequencing.operation.Operation` (*uuid=None, params=None, status="PENDING", result=None, metadata=None*)

In fact it's a `kser.sequencing.task.Task` which has other task as child.

**classmethod** `new` (*\*\*kwargs*)

**Warning:** Deprecated, do not use this method anymore.

Initialize the operation using a dict.

**Parameters** **kwargs** (*dict*) – key arguments

**Returns** A new operation

**Return type** `kser.sequencing.operation.Operation`

**classmethod** `parse_inputs (**kwargs)`

**Warning:** Deprecated, do not use this method anymore.

Use by `kser.sequencing.operation.Operation.new` to check inputs

**Parameters** `kwargs (dict)` – key arguments

**Returns** parsed input

**Return type** dict

**set\_status** (`status, result=None`)

Update operation status

**Parameters**

- **status** (`str`) – New status
- **result** (`cdumay_result.Result`) – Execution result

**add\_task** (`task`)

Add task to operation

**Parameters** `task (kser.sequencing.task.Task)` – task to add

**prebuild** (`**kwargs`)

To implement, perform check before the operation creation

**Parameters** `kwargs (dict)` – key arguments

**next** (`task`)

Find the next task

**Parameters** `task (kser.sequencing.task.Task)` – previous task

**Returns** The next task

**Return type** `kser.sequencing.task.Task` or None

**launch\_next** (`task=None, result=None`)

Launch next task or finish operation

**Parameters**

- **task** (`kser.sequencing.task.Task`) – previous task
- **result** (`cdumay_result.Result`) – previous task result

**Returns** Execution result

**Return type** `cdumay_result.Result`

**launch** ()

Send the first task

**Returns** Execution result

**Return type** `cdumay_result.Result`

**finalize** ()

To implement, post build actions (database mapping ect...)

**Returns** Self return

**Return type** *kser.sequencing.operation.Operation*

**build\_tasks** (*\*\*kwargs*)

Initialize tasks

**Parameters** **kwargs** (*dict*) – tasks parameters (~=context)

**Returns** list of tasks

**Return type** list(*kser.sequencing.task.Task*)

**compute\_tasks** (*\*\*kwargs*)

Perfrom checks and build tasks

**Returns** list of tasks

**Return type** list(*kser.sequencing.operation.Operation*)

**build** (*\*\*kwargs*)

Create the operation and associate tasks

**Parameters** **kwargs** (*dict*) – operation data

**Returns** The operation

**Return type** *kser.sequencing.operation.Operation*

**send** ()

To implement, send operation to Kafka

**Returns** The operation

**Return type** *kser.sequencing.operation.Operation*

**class** *kser.sequencing.registry.OperationRegistry* (*app=None*, *controller\_class=kser.controller.Controller*)  
A which route *kser.schemas.Message* from Kafka to the requested *kser.sequencing.operation.Operation*.

**Parameters**

- **app** (*flask.Flask*) – Flask application if any
- **controller\_class** (*kser.controller.Controller*) – Controller to use

**subscribe** (*callback*)

Register an *kser.sequencing.operation.Operation* into the controller. This method is a shortcut to *kser.controller.Controller.register*.

**Parameters** **callback** (*kser.sequencing.task.Task*) – Any class which implement *Task*.

**load\_tasks** ()

To implement, load operation tasks

### 1.3.2 Example

The following example is based on a dice game, player roll tree time dices.

#### Consumer

```

1  import logging
2  logging.basicConfig(
3      level=logging.INFO, format="%asctime)s %(levelname)-8s %(message)s"
4  )
5
6  import random
7
8  from cdumay_result import Result
9  from kser.sequencing.operation import Operation
10 from kser.sequencing.registry import OperationRegistry
11 from kser.sequencing.task import Task
12
13 oregistry = OperationRegistry()
14
15
16 @oregistry.subscribe
17 class DiceRoll(Task):
18     def run(self):
19         launch = random.randint(1, 6)
20         return Result(uuid=self.uuid, stdout="You made a {}".format(launch))
21
22
23 @oregistry.subscribe
24 class DiceLaunch(Operation):
25     def build_tasks(self, **kwargs):
26         return [DiceRoll(), DiceRoll(), DiceRoll()]
27
28
29 if __name__ == '__main__':
30     from flask import Flask
31     from kser.python_kafka.consumer import Consumer
32
33     app = Flask(__name__)
34     oregistry.init_app(app)
35     cons = Consumer(...)
36     cons.REGISTER = oregistry.controller
37     cons.run()

```

**Explanations:**

- **line 13:** We initialize the registry
- **line 16/23:** We subscribe the task/operation into the registry
- **line 35-37:** We start the consumer

**Producer**

Producer has nothing special for this feature.

```

1  import logging
2  logging.basicConfig(
3      level=logging.INFO, format="%asctime)s %(levelname)-8s %(message)s"
4  )
5
6  import uuid
7  from kser.python_kafka.producer import Producer

```

(continues on next page)

(continued from previous page)

```

8  from kser.schemas import Message
9
10 pro = Producer(...)
11 pro.send('...', Message(uuid=str(uuid.uuid4()), entrypoint="__main__.DiceRoll"))

```

**Explanations:**

- **line 10:** We initialize the producer
- **line 11:** We send a Message with the entrypoint `__main__.DiceRoll` that matches our Task registered

**Execution****Producer console output:**

```

2018-08-10 18:46:20,082 INFO      <BrokerConnection host=*****/*****_
↳port=9093>: Authenticated as admin
2018-08-10 18:46:20,549 INFO      Broker version identified as 0.10
2018-08-10 18:46:20,550 INFO      Set configuration api_version=(0, 10) to skip auto_
↳check_version requests on startup
2018-08-10 18:46:20,711 INFO      <BrokerConnection host=*****/*****_
↳port=9093>: Authenticated as admin
2018-08-10 18:46:20,731 INFO      Producer.Success: __main__.DiceRoll[872c3be0-51ac-
↳457f-a4d9-12c2f7667b80]: Message __main__.DiceRoll[872c3be0-51ac-457f-a4d9-
↳12c2f7667b80] sent in *****
2018-08-10 18:46:20,731 INFO      Closing the Kafka producer with 999999999 secs_
↳timeout.
2018-08-10 18:46:20,741 INFO      Kafka producer closed

```

**Consumer console output:**

```

2018-08-10 18:44:42,355 INFO      Operation registry: loaded __main__.DiceRoll
2018-08-10 18:44:42,355 INFO      Operation registry: loaded __main__.DiceLaunch
2018-08-10 18:44:42,696 INFO      <BrokerConnection host=*****/*****_
↳port=9093>: Authenticated as admin
2018-08-10 18:44:43,163 INFO      Broker version identified as 0.10
2018-08-10 18:44:43,163 INFO      Set configuration api_version=(0, 10) to skip auto_
↳check_version requests on startup
2018-08-10 18:44:43,164 INFO      Updating subscribed topics to: ['*****']
2018-08-10 18:44:43,165 INFO      Consumer.Starting...
2018-08-10 18:44:43,182 INFO      Group coordinator for ***** is_
↳BrokerMetadata(nodeId=114251126, host='*****', port=9093, rack=None)
2018-08-10 18:44:43,182 INFO      Discovered coordinator 114251126 for group *****
2018-08-10 18:44:43,182 INFO      Revoking previously assigned partitions set() for_
↳group *****
2018-08-10 18:44:43,182 INFO      (Re-)joining group *****
2018-08-10 18:44:46,230 INFO      Skipping heartbeat: no auto-assignment or waiting on_
↳rebalance
2018-08-10 18:44:46,249 INFO      Joined group '*****' (generation 2) with member_id_
↳kafka-python-1.3.1-db5cdcc7-3be9-4cf6-a4e7-06a97cc69120
2018-08-10 18:44:46,249 INFO      Elected group leader -- performing partition_
↳assignments using range
2018-08-10 18:44:46,268 INFO      Successfully joined group ***** with generation 2
2018-08-10 18:44:46,268 INFO      Updated partition assignment: [TopicPartition(topic=
↳'*****', partition=0), TopicPartition(topic='*****', partition=1),_
↳TopicPartition(topic='*****', partition=2)]

```

(continues on next page)



(continued from previous page)

```

2018-08-10 18:44:46,269 INFO      Setting newly assigned partitions
→{TopicPartition(topic='*****', partition=0), TopicPartition(topic='*****',
→partition=1), TopicPartition(topic='*****', partition=2)} for group *****
2018-08-10 18:46:20,750 INFO      DiceRoll.Success: __main__.DiceRoll[872c3be0-51ac-
→457f-a4d9-12c2f7667b80]: You made a 1
2018-08-10 18:46:20,751 INFO      Controller.Success: __main__.DiceRoll[872c3be0-51ac-
→457f-a4d9-12c2f7667b80]: You made a 1

```

As we can see, the task `__main__.DiceRoll` is sent by the producer and executed by the consumer with the stdout “You made a 1”

## 1.4 Prometheus export

Kser support [prometheus](#) metric export.

### 1.4.1 Install

```
$ pip install kser[prometheus]
```

### 1.4.2 Configuration

Configuration is done using environment variable:

Table 1: Configuration variables

Environment variable	Default value
KSER_METRICS_ENABLED	no
KSER_METRICS_ADDRESS	0.0.0.0
KSER_METRICS_PORT	8888

The exporter has only 2 metrics defined by default, it’s just a sample. A good way to implement your own is to override the triggers methods (prefixed with ‘\_’) like the following example:

```

1  from kser import KSER_METRICS_ENABLED
2  from prometheus_client import Counter
3  from kser.entry import Entrypoint
4
5  MY_METRIC = Counter('kser_my_metric', 'a usefull metric')
6
7
8  class MyEntrypoint(Entrypoint):
9      def _run(self):
10         if KSER_METRICS_ENABLED == "yes":
11             MY_METRIC.inc()
12
13         return self.run()

```

See also:

[prometheus\\_client](#) Prometheus instrumentation library for Python applications.

## 1.5 Opentracing support with Jaeger

Kser support [opentracing](#) and [Jaeger](#) to follow and display operation and task execution.

### 1.5.1 Install

```
$ pip install kser[opentracing]
```

### 1.5.2 Configuration

Configuration is done using environment variable:

Table 2: Configuration variable

Environment variable	Default value
JAEGER_HOST	localhost

### 1.5.3 Example

---

**Note:** We assume that you have a working jaeger...

---

The following example is based on a dice game, player roll five time dices:

```
1 import logging
2
3 logging.basicConfig(
4     level=logging.DEBUG,
5     format="%(asctime)s %(levelname)-8s %(message)s"
6 )
7
8 import time
9 import random
10
11 from cdumay_result import Result
12 from kser.tracing.task import OpentracingTask
13 from kser.tracing.operation import OpentracingOperation
14
15
16 def contrived_time():
17     time.sleep(random.randint(1, 20) / 10)
18
19
20 class DiceRoll(OpentracingTask):
21     def prerun(self):
22         contrived_time()
23
24     def run(self):
25         launch = random.randint(1, 6)
26         time.sleep(random.randint(1, 3))
27         return Result(uuid=self.uuid, stdout="You made a {}".format(launch))
28
```

(continues on next page)

(continued from previous page)

```

29     def postrun(self, result):
30         contrived_time()
31         return result
32
33
34 class DiceLaunch(OpenTracingOperation):
35     def build_tasks(self, **kwargs):
36         return [
37             DiceRoll(),
38             DiceRoll(),
39             DiceRoll(),
40         ]
41
42
43 if __name__ == '__main__':
44     import os
45     from jaeger_client import Config
46     from cdumay_opentracing import OpenTracingManager
47     from kser.entry import Entrypoint
48     from kser.tracing.driver import OpenTracingKserDriver
49
50     tracer = Config(service_name="test-kser", config=dict(
51         sampler=dict(type='const', param=1), logging=True,
52         local_agent=dict(reporting_host=os.getenv('JAEGER_HOST', 'localhost'))
53     )).initialize_tracer()
54
55     OpenTracingManager.register(Entrypoint, OpenTracingKserDriver)
56
57     DiceLaunch().build().execute()
58     time.sleep(4)
59     tracer.close()

```

**Explanations:**

- **line 11-12:** as we can see, we use the specialized classes `kser.tracing.operation.OpenTracingOperation` and `kser.tracing.task.OpenTracingTask` based on `kser.sequencing.operation.Operation` and `kser.sequencing.task.Task`.
- **line 21/25/29:** we simulate execution time
- **line 50-53:** We initialize tracing using Jaeger.
- **line 55:** We register the Kser driver `kser.tracing.driver.OpenTracingKserDriver` for subclass of `kser.entry.Entrypoint`.
- **line 57:** We launch the operation `DiceLaunch`.
- **line 59:** yield to `IOLoop` to flush the spans
- **line 60:** we flush any buffered spans

**Console output:**

```

2018-08-08 12:36:59,753 Initializing Jaeger Tracer with UDP reporter
2018-08-08 12:36:59,754 Using sampler ConstSampler(True)
2018-08-08 12:36:59,758 opentracing.tracer initialized to <jaeger_client.tracer.
↪Tracer object at 0x7f1c7d59a668>[app_name=kser]
2018-08-08 12:36:59,759 DiceLaunch.PreBuild: __main__.DiceLaunch[8a75d3f6-71b6-4dad-
↪98b5-faled6b2f160]: {}

```

(continues on next page)

(continued from previous page)

```

2018-08-08 12:36:59,760 DiceLaunch.BuildTasks: 5 task(s) found
2018-08-08 12:36:59,760 DiceLaunch[8a75d3f6-71b6-4dad-98b5-faled6b2f160] - PreRun
2018-08-08 12:36:59,760 DiceLaunch.SetStatus: __main__.DiceLaunch[8a75d3f6-71b6-4dad-
↳98b5-faled6b2f160] status update 'PENDING' -> 'RUNNING'
2018-08-08 12:36:59,760 Reporting span↳
↳b72e015e7d14cf41:9ae003a32eafcc52:b72e015e7d14cf41:1 kser.DiceLaunch[8a75d3f6-71b6-
↳4dad-98b5-faled6b2f160] - PreRun
2018-08-08 12:36:59,761 DiceRoll[bb9e2fa6-29a9-4431-95ee-6573103349c5] - PreRun
2018-08-08 12:37:01,462 Reporting span↳
↳b72e015e7d14cf41:411604082021b77a:3478b474ed615e6:1 kser.DiceRoll[bb9e2fa6-29a9-
↳4431-95ee-6573103349c5] - PreRun
2018-08-08 12:37:01,462 DiceRoll[bb9e2fa6-29a9-4431-95ee-6573103349c5] - Run
2018-08-08 12:37:02,463 Reporting span↳
↳b72e015e7d14cf41:af5d7ba4127a6f6c:3478b474ed615e6:1 kser.DiceRoll[bb9e2fa6-29a9-
↳4431-95ee-6573103349c5] - Run
2018-08-08 12:37:02,463 DiceRoll[bb9e2fa6-29a9-4431-95ee-6573103349c5] - PostRun
2018-08-08 12:37:03,865 Reporting span↳
↳b72e015e7d14cf41:cb0a705e20b36496:3478b474ed615e6:1 kser.DiceRoll[bb9e2fa6-29a9-
↳4431-95ee-6573103349c5] - PostRun
2018-08-08 12:37:03,865 DiceRoll.Success: __main__.DiceRoll[bb9e2fa6-29a9-4431-95ee-
↳6573103349c5]: You made a 3
2018-08-08 12:37:03,865 Reporting span↳
↳b72e015e7d14cf41:3478b474ed615e6:b72e015e7d14cf41:1 kser.DiceRoll[bb9e2fa6-29a9-
↳4431-95ee-6573103349c5] - Execute
2018-08-08 12:37:03,866 DiceRoll[31090ded-0563-4497-ac7f-7523f79c4bb9] - PreRun
2018-08-08 12:37:04,066 Reporting span↳
↳b72e015e7d14cf41:c49fb0550bd27b25:1ec1d003a3d94c66:1 kser.DiceRoll[31090ded-0563-
↳4497-ac7f-7523f79c4bb9] - PreRun
2018-08-08 12:37:04,067 DiceRoll[31090ded-0563-4497-ac7f-7523f79c4bb9] - Run
2018-08-08 12:37:05,067 Reporting span↳
↳b72e015e7d14cf41:1be20968f7d6cc1f:1ec1d003a3d94c66:1 kser.DiceRoll[31090ded-0563-
↳4497-ac7f-7523f79c4bb9] - Run
2018-08-08 12:37:05,068 DiceRoll[31090ded-0563-4497-ac7f-7523f79c4bb9] - PostRun
2018-08-08 12:37:06,870 Reporting span↳
↳b72e015e7d14cf41:70c9928dd1f6e1df:1ec1d003a3d94c66:1 kser.DiceRoll[31090ded-0563-
↳4497-ac7f-7523f79c4bb9] - PostRun
2018-08-08 12:37:06,871 DiceRoll.Success: __main__.DiceRoll[31090ded-0563-4497-ac7f-
↳7523f79c4bb9]: You made a 3
2018-08-08 12:37:06,871 Reporting span↳
↳b72e015e7d14cf41:1ec1d003a3d94c66:b72e015e7d14cf41:1 kser.DiceRoll[31090ded-0563-
↳4497-ac7f-7523f79c4bb9] - Execute
2018-08-08 12:37:06,871 DiceRoll[1d1a0361-fe0f-4cbe-92fc-abe0a801dbal] - PreRun
2018-08-08 12:37:07,072 Reporting span↳
↳b72e015e7d14cf41:f337f69f34da1345:c17278695688fe0d:1 kser.DiceRoll[1d1a0361-fe0f-
↳4cbe-92fc-abe0a801dbal] - PreRun
2018-08-08 12:37:07,072 DiceRoll[1d1a0361-fe0f-4cbe-92fc-abe0a801dbal] - Run
2018-08-08 12:37:10,075 Reporting span↳
↳b72e015e7d14cf41:304bef5db6be72c2:c17278695688fe0d:1 kser.DiceRoll[1d1a0361-fe0f-
↳4cbe-92fc-abe0a801dbal] - Run
2018-08-08 12:37:10,076 DiceRoll[1d1a0361-fe0f-4cbe-92fc-abe0a801dbal] - PostRun
2018-08-08 12:37:11,677 Reporting span↳
↳b72e015e7d14cf41:9499a64a4c55ebf:c17278695688fe0d:1 kser.DiceRoll[1d1a0361-fe0f-
↳4cbe-92fc-abe0a801dbal] - PostRun
2018-08-08 12:37:11,677 DiceRoll.Success: __main__.DiceRoll[1d1a0361-fe0f-4cbe-92fc-
↳abe0a801dbal]: You made a 5
2018-08-08 12:37:11,677 Reporting span↳
↳b72e015e7d14cf41:c17278695688fe0d:b72e015e7d14cf41:1 kser.DiceRoll[1d1a0361-fe0f-
↳4cbe-92fc-abe0a801dbal] - Execute

```

(continues on next page)

(continued from previous page)

```

2018-08-08 12:37:11,678 DiceRoll[f24e7816-e953-4183-891c-a3a3dc008128] - PreRun
2018-08-08 12:37:12,579 Reporting span
↳b72e015e7d14cf41:d2200a6f5b029a7f:942840d8feffd756:1 kser.DiceRoll[f24e7816-e953-
↳4183-891c-a3a3dc008128] - PreRun
2018-08-08 12:37:12,580 DiceRoll[f24e7816-e953-4183-891c-a3a3dc008128] - Run
2018-08-08 12:37:15,580 Reporting span
↳b72e015e7d14cf41:139b24d83aee44bf:942840d8feffd756:1 kser.DiceRoll[f24e7816-e953-
↳4183-891c-a3a3dc008128] - Run
2018-08-08 12:37:15,581 DiceRoll[f24e7816-e953-4183-891c-a3a3dc008128] - PostRun
2018-08-08 12:37:16,782 Reporting span
↳b72e015e7d14cf41:95fdae9ade3b3f7:942840d8feffd756:1 kser.DiceRoll[f24e7816-e953-
↳4183-891c-a3a3dc008128] - PostRun
2018-08-08 12:37:16,783 DiceRoll.Success: __main__.DiceRoll[f24e7816-e953-4183-891c-
↳a3a3dc008128]: You made a 4
2018-08-08 12:37:16,783 Reporting span
↳b72e015e7d14cf41:942840d8feffd756:b72e015e7d14cf41:1 kser.DiceRoll[f24e7816-e953-
↳4183-891c-a3a3dc008128] - Execute
2018-08-08 12:37:16,784 DiceRoll[ae5d36a7-a2b7-4ac1-a5e8-39004059e3c5] - PreRun
2018-08-08 12:37:18,686 Reporting span
↳b72e015e7d14cf41:bea78e6f943f7aa8:54941f4bb4657ec2:1 kser.DiceRoll[ae5d36a7-a2b7-
↳4ac1-a5e8-39004059e3c5] - PreRun
2018-08-08 12:37:18,687 DiceRoll[ae5d36a7-a2b7-4ac1-a5e8-39004059e3c5] - Run
2018-08-08 12:37:21,690 Reporting span
↳b72e015e7d14cf41:fc937aa7e2d633d3:54941f4bb4657ec2:1 kser.DiceRoll[ae5d36a7-a2b7-
↳4ac1-a5e8-39004059e3c5] - Run
2018-08-08 12:37:21,691 DiceRoll[ae5d36a7-a2b7-4ac1-a5e8-39004059e3c5] - PostRun
2018-08-08 12:37:22,892 Reporting span
↳b72e015e7d14cf41:c4393051442fdecc:54941f4bb4657ec2:1 kser.DiceRoll[ae5d36a7-a2b7-
↳4ac1-a5e8-39004059e3c5] - PostRun
2018-08-08 12:37:22,893 DiceRoll.Success: __main__.DiceRoll[ae5d36a7-a2b7-4ac1-a5e8-
↳39004059e3c5]: You made a 6
2018-08-08 12:37:22,893 Reporting span
↳b72e015e7d14cf41:54941f4bb4657ec2:b72e015e7d14cf41:1 kser.DiceRoll[ae5d36a7-a2b7-
↳4ac1-a5e8-39004059e3c5] - Execute
2018-08-08 12:37:22,894 DiceLaunch[8a75d3f6-71b6-4dad-98b5-faled6b2f160] - PostRun
2018-08-08 12:37:22,894 Reporting span
↳b72e015e7d14cf41:d31e32e75feaba57:b72e015e7d14cf41:1 kser.DiceLaunch[8a75d3f6-71b6-
↳4dad-98b5-faled6b2f160] - PostRun
2018-08-08 12:37:22,894 DiceLaunch.SetStatus: __main__.DiceLaunch[8a75d3f6-71b6-4dad-
↳98b5-faled6b2f160] status update 'RUNNING' -> 'SUCCESS'
2018-08-08 12:37:22,895 DiceLaunch.Success: __main__.DiceLaunch[8a75d3f6-71b6-4dad-
↳98b5-faled6b2f160]: You made a 6
2018-08-08 12:37:22,895 Reporting span b72e015e7d14cf41:b72e015e7d14cf41:0:1 kser.
↳DiceLaunch[8a75d3f6-71b6-4dad-98b5-faled6b2f160] - Execute

```

**Display in jaeger:**

Select **kser** in the service list and click on you trace:



**Note:** Only execution is traced

### Deeper:

Kser allow you to create operation using another operation using `kser.tracing.operation`. `OpentracingOperation.compute_tasks()`. Computed tasks will be append to the operation span directly. The reuse the the previous code and append

```
1 class Yahtzee(OpentracingOperation):
2     def build_tasks(self, **kwargs):
3         tasks = list()
4         tasks += DiceLaunch().compute_tasks()
5         tasks += DiceLaunch().compute_tasks()
6         tasks += DiceLaunch().compute_tasks()
7         return tasks
```

This will create a span with 3x5 tasks in it.

### See also:

**cdumay-opentracing** Library to facilitate opentracing integration

**OpenTracing API for Python** Python library for OpenTracing.

**jaeger-client-python** Jaeger Bindings for Python OpenTracing API

## A

`add_task()` (kser.sequencing.operation.Operation method), 9

## B

`build()` (kser.sequencing.operation.Operation method), 10  
`build_tasks()` (kser.sequencing.operation.Operation method), 10

## C

`check_required_params()` (kser.entry.Entrypoint method), 3  
`compute_tasks()` (kser.sequencing.operation.Operation method), 10

## D

`decode()` (kser.crypto.CryptoMessage method), 6

## E

`encode()` (kser.crypto.CryptoMessage method), 6  
`execute()` (kser.entry.Entrypoint method), 3

## F

`finalize()` (kser.sequencing.operation.Operation method), 9  
`from_Message()` (kser.entry.Entrypoint class method), 3

## K

`kser.entry.Entrypoint` (built-in class), 3  
`kser.sequencing.operation.Operation` (built-in class), 8  
`kser.sequencing.registry.OperationRegistry` (built-in class), 10  
`kser.sequencing.task.Task` (built-in class), 8

## L

`launch()` (kser.sequencing.operation.Operation method), 9  
`launch_next()` (kser.sequencing.operation.Operation method), 9

`load_tasks()` (kser.sequencing.registry.OperationRegistry method), 10

`log()` (kser.entry.Entrypoint method), 3  
`log()` (kser.sequencing.task.Task method), 8

## N

`new()` (kser.sequencing.operation.Operation class method), 8  
`next()` (kser.sequencing.operation.Operation method), 9

## O

`onerror()` (kser.entry.Entrypoint method), 4  
`onsuccess()` (kser.entry.Entrypoint method), 4

## P

`parse_inputs()` (kser.sequencing.operation.Operation class method), 8  
`postinit()` (kser.entry.Entrypoint method), 4  
`postrun()` (kser.entry.Entrypoint method), 4  
`prebuild()` (kser.sequencing.operation.Operation method), 9  
`prerun()` (kser.entry.Entrypoint method), 4

## R

`REQUIRED_FIELDS` (kser.entry.Entrypoint attribute), 3  
`run()` (kser.entry.Entrypoint method), 4

## S

`send()` (kser.sequencing.operation.Operation method), 10  
`set_status()` (kser.sequencing.operation.Operation method), 9  
`subscribe()` (kser.sequencing.registry.OperationRegistry method), 10

## T

`to_Message()` (kser.entry.Entrypoint method), 4

## U

`unsafe_execute()` (kser.entry.Entrypoint method), 4