mesoshttp Documentation

Release 0.1.0

Olivier Sallou

Nov 07, 2018

Contents

This is a Python library to create a Mesos scheduler.

It removes the need to use/install Mesos python bindings and makes use of the HTTP API (native bindings are not updated anymore with new features)

MesosClient is the main entrypoint that connects to the Mesos master. You can then register for callbacks on events. On subscribe, you get a driver instance which you can use to send messages to master on current framework (kill, etc.)

MesosClient must be executed in a separate thread as it keeps a loop pooling connection with the master.

Submitted tasks should be in JSON format (according to mesos.proto).

See sample/test.py for example.

Callbacks will "block" the mesos message treatment, so they should be short, or messages should be forwarded to a queue in an other thread/process where longer tasks will handle messages.

Contents:

CHAPTER 1

MesosClient

1.1 MesosClient reference

class mesoshttp.client.**MesosClient**(*mesos_urls*, *frameworkId=None*, *frameworkName='Mesos HTTP framework'*, *frameworkUser='root'*, *frameworkHostname=''*, *frameworkWebUI=''*, *max_reconnect=3*)

Entrypoint class to connect framework to Mesos master

Instance should be started in a separate thread as *MesosClient.register* will start a blocking loop with a long connection.

class SchedulerDriver(*mesos_url*, *frameworkId*, *streamId*, *requests_auth=None*, *verify=True*)

Handler to communicate with scheduler

MesosClient.SchedulerDriver instance is available after the SUBSCRIBED event with the subscribed event.

__init__(*mesos_url*, *frameworkId*, *streamId*, *requests_auth=None*, *verify=True*)

Create a driver instance related to created framework

kill(*agent_id*, *task_id*)

Kill specified task

Parameters

- **agent_id** (*str*) – slave agent_id
- **task_id** (*str*) – task identifier

message(*agent_id*, *executor_id*, *message*)

Send message to an executor

Parameters

- **agent_id** (*str*) – slave identifier
- **executor_id** (*str*) – executor identifier
- **message** (*str*) – message to send, raw bytes encoded as Base64

reconcile(*tasks*)
Reconcile tasks
Parameters **tasks** (*list*) – list of dict { "agent_id": xx, "task_id": yy }

request(*requests*)
Send a REQUEST message
Parameters **requests** (*list*) – list of resources request [{'agent_id': : XX, 'resources': {}}]

revive()
Send REVIVE request

shutdown(*agent_id*, *executor_id*)
Shutdown an executor
Parameters
- **agent_id** (*str*) – slave identifier
- **executor_id** (*str*) – executor identifier

tearDown()
Undeclare framework

_MesosClient__zk_detect(*zk_url*, *prefix='/mesos'*)
Try to get master url info from zookeeper

Parameters
- **zk_url** (*str*) – ip/port to reach zookeeper
- **prefix** (*str*) – prefix to search for in zookeeper

__init__(*mesos_urls*, *frameworkId=None*, *frameworkName='Mesos HTTP framework'*, *frameworkUser='root'*, *frameworkHostname=''*, *frameworkWebUI=''*, *max_reconnect=3*)
Create a frameworkId

Parameters
- **mesos_urls** (*list*) – list of mesos http endpoints
- **frameworkId** (*str*) – identifier of the framework, if None, will declare a new framework
- **frameworkName** (*str*) – name of the framework
- **frameworkUser** (*str*) – user to use (will run tasks as user), defaults to root
- **frameworkHostname** (*str*) – Hostname of the framework
- **frameworkWebUI** (*str*) – URL of the framework WebUI
- **max_reconnect** (*int defaults to 3*) – number of reconnection retries when connection fails

__weakref__
list of weak references to the object (if defined)

add_capability(*capability*)
Adds a framwork capability

Parameters **capability** (*str*) – caapbility name

combine_offers(*offers*, *operations*, *options=None*)
Accept offers with task operations

Parameters

- **offers** (*list*) – offers to be accepted
- **operations** (*list of json TaskInfo*) – JSON TaskInfo instances to accept
- **options** (*JSON filters instances*) – optional filters

This method does not check if the operations are valid JSON TaskInfo instances and if conform with the offers.

disconnect_framework()
Stops framework but does not teardown (unregister) the framework

This will enable framework reconnection and will not kill running jobs

get_driver()
Get driver instance to dialog with master

Returns *MesosClient.SchedulerDriver*

get_master_info()
Get Mesos master information, return None if not connected

Returns json formatted info about connected Mesos master

on(*eventName*, *callback*)
Register callback for an event.

Multiple callbacks can be registered for the same event

Parameters

- **eventName** (*str*) – name fo the event to register to (*MesosClient.SUBSCRIBED*, etc.)
- **callback** (*def*) – function to call on event

register()
Register framework, return False if could not connect, else will open a permanent HTTP connection.

Creates an infinite loop on a permanent connection to Mesos master to receive messages. On message, callbacks will be called.

set_checkpoint(*do_checkpoint*)
Sets framework checkpoint value

Parameters **do_checkpoint** (*bool*) – de/activate checkpoint in framework

set_credentials(*principal*, *secret*)
Set credentials to authenticate with Mesos master

Parameters

- **principal** (*str*) – login to use
- **secret** (*str*) – password

set_failover_timeout(*timeout*)
Sets failover timeout value

Parameters **timeout** (*int*) – define framework failover timeout, in seconds

set_role(*role_name*)
Set Mesos role to use by framework

Parameters **role_name** (*str*) – Mesos role name

set_service_account(*service_secret*, *verify=False*)
Set credentials to authenticate with DCOS and Mesos Master

Parameters

- **service_secret** (*dict*) – Optional DCOS Service account secret. Supersedes principal / secret.
- **verify** (*bool*) – validate HTTPS fronted Mesos API using CA root trusts, defaults to False

tearDown()
Unregister and stop scheduler

CHAPTER 2

Offers

2.1 Offers reference

class mesoshttp.offers.**Offer**(*mesos_url*, *frameworkId*, *streamId*, *mesosOffer*, *requests_auth=None*, *verify=True*)

Wrapper class for Mesos offers

__init__(*mesos_url*, *frameworkId*, *streamId*, *mesosOffer*, *requests_auth=None*, *verify=True*)

Initialize self. See help(type(self)) for accurate signature.

accept(*operations*, *options=None*)

Accept offer with task operations

Parameters

- **operations** (*list of json TaskInfo*) – JSON TaskInfo instances to accept in current offer
- **options** (*dict*) – Optional offer additional params (filters, ...)

decline(*options=None*)

Decline offer

Parameters **options** (*dict*) – Optional offer additional params (filters, ...)

get_offer()

Get offer info received from Mesos

Returns dict

CHAPTER 3

Update

3.1 Update reference

class mesoshttp.update.**Update**(*mesos_url*, *frameworkId*, *streamId*, *mesosUpdate*, *requests_auth=None*, *verify=True*)

This class manages Update message from Mesos master

__init__(*mesos_url*, *frameworkId*, *streamId*, *mesosUpdate*, *requests_auth=None*, *verify=True*)

Initialize self. See help(type(self)) for accurate signature.

ack()

Acknowledge an update message

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

Symbols

A

C

D

G

K

M

O

R

S

T

U