pylivy Documentation

Andrew Crozier

May 28, 2020

CONTENTS

Livy is an open source REST interface for interacting with Spark. `pylivy` is a Python client for Livy, enabling easy remote code execution on a Spark cluster.

CHAPTER

ONE

INSTALLATION

```
$ pip install -U livy
```

Note that `pylivy` requires Python 3.6 or later.

CHAPTER

TWO

USAGE

The *LivySession* class is the main interface provided by `pylivy`:

```
from livy import LivySession

LIVY_URL = 'http://spark.example.com:8998'

with LivySession(LIVY_URL) as session:
    # Run some code on the remote cluster
    session.run("filtered = df.filter(df.name == 'Bob')")
    # Retrieve the result
    local_df = session.read('filtered')
```

Authenticate requests sent to Livy by passing any requests Auth object to the `LivySession`. For example, to perform HTTP basic auth do:

```
from requests.auth import HTTPBasicAuth

auth = HTTPBasicAuth('username', 'password')

with LivySession(LIVY_URL, auth) as session:
    session.run("filtered = df.filter(df.name == 'Bob')")
    local_df = session.read('filtered')
```

CHAPTER

THREE

API DOCUMENATION

3.1 livy.session

class livy.session.**LivySession**(*url*, *auth=None*, *kind=<SessionKind.PYSPARK: 'pyspark'>*, *proxy_user=None*, *spark_conf=None*, *echo=True*, *check=True*)

Manages a remote Livy session and high-level interactions with it.

Parameters

- **url** (str) – The URL of the Livy server.
- **kind** (SessionKind) – The kind of session to create.
- **proxy_user** (Optional[str]) – User to impersonate when starting the session.
- **spark_conf** (Optional[Dict[str, Any]]) – Spark configuration properties.
- **echo** (bool) – Whether to echo output printed in the remote session. Defaults to True.
- **check** (bool) – Whether to raise an exception when a statement in the remote session fails. Defaults to True.

start()

Create the remote Spark session and wait for it to be ready.

Return type None

property state

The state of the managed Spark session.

Return type SessionState

close()

Kill the managed Spark session.

Return type None

run(*code*)

Run some code in the managed Spark session.

Parameters **code** (str) – The code to run.

Return type Output

read(*dataframe_name*)

Evaluate and retrieve a Spark dataframe in the managed session.

Parameters **dataframe_name** (str) – The name of the Spark dataframe to read.

Return type DataFrame

read_sql(*code*)
Evaluate a Spark SQL satatement and retrieve the result.

Parameters **code** (str) – The Spark SQL statement to evaluate.

Return type DataFrame

3.2 livy.client

class livy.client.**LivyClient**(*url*, *auth=None*)
A client for sending requests to a Livy server.

Parameters

- **url** (str) – The URL of the Livy server.
- **auth** (Union[AuthBase, Tuple[str, str], None]) – A requests-compatible auth object to use when making requests.

close()
Close the underlying requests session.

Return type None

server_version()
Get the version of Livy running on the server.

Return type Version

legacy_server()
Determine if the server is running a legacy version.

Legacy versions support different session kinds than newer versions of Livy.

Return type bool

list_sessions()
List all the active sessions in Livy.

Return type List[Session]

create_session(*kind*, *proxy_user=None*, *spark_conf=None*)
Create a new session in Livy.

Parameters

- **kind** (SessionKind) – The kind of session to create.
- **proxy_user** (Optional[str]) – User to impersonate when starting the session.
- **spark_conf** (Optional[Dict[str, Any]]) – Spark configuration properties.

Return type Session

get_session(*session_id*)
Get information about a session.

Parameters **session_id** (int) – The ID of the session.

Return type Optional[Session]

delete_session(*session_id*)
Kill a session.

Parameters **session_id** (int) – The ID of the session.

Return type None

list_statements(*session_id*)
Get all the statements in a session.

Parameters **session_id** (int) – The ID of the session.

Return type List[Statement]

create_statement(*session_id*, *code*, *kind=None*)
Run a statement in a session.

Parameters

- **session_id** (int) – The ID of the session.
- **code** (str) – The code to execute.
- **kind** (Optional[StatementKind]) – The kind of code to execute.

Return type Statement

get_statement(*session_id*, *statement_id*)
Get information about a statement in a session.

Parameters

- **session_id** (int) – The ID of the session.
- **statement_id** (int) – The ID of the statement.

Return type Statement

PYTHON MODULE INDEX

INDEX