# 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*, *jars=None*, *py_files=None*, *files=None*, *driver_memory=None*, *driver_cores=None*, *executor_memory=None*, *executor_cores=None*, *num_executors=None*, *archives=None*, *queue=None*, *name=None*, *spark_conf=None*, *echo=True*, *check=True*)

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

The py_files, files, jars and archives arguments are lists of URLs, e.g. ["s3://bucket/object", "hdfs://path/to/file", ...] and must be reachable by the Spark driver process. If the provided URL has no scheme, it's considered to be relative to the default file system configured in the Livy server.

URLs in the py_files argument are copied to a temporary staging area and inserted into Python's sys.path ahead of the standard library paths. This allows you to import .py, .zip and .egg files in Python.

URLs for jars, py_files, files and archives arguments are all copied to the same working directory on the Spark cluster.

The driver_memory and executor_memory arguments have the same format as JVM memory strings with a size unit suffix ("k", "m", "g" or "t") (e.g. 512m, 2g).

See https://spark.apache.org/docs/latest/configuration.html for more information on Spark configuration properties.

**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.
- **jars** (Optional[List[str]]) – URLs of jars to be used in this session.
- **py_files** (Optional[List[str]]) – URLs of Python files to be used in this session.
- **files** (Optional[List[str]]) – URLs of files to be used in this session.
- **driver_memory** (Optional[str]) – Amount of memory to use for the driver process (e.g. '512m').
- **driver_cores** (Optional[int]) – Number of cores to use for the driver process.
- **executor_memory** (Optional[str]) – Amount of memory to use per executor process (e.g. '512m').

- **executor_cores** (Optional[int]) – Number of cores to use for each executor.
- **num_executors** (Optional[int]) – Number of executors to launch for this session.
- **archives** (Optional[List[str]]) – URLs of archives to be used in this session.
- **queue** (Optional[str]) – The name of the YARN queue to which submitted.
- **name** (Optional[str]) – The name of this 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*, *jars=None*, *py_files=None*, *files=None*, *driver_memory=None*, *driver_cores=None*, *executor_memory=None*, *executor_cores=None*, *num_executors=None*, *archives=None*, *queue=None*, *name=None*, *spark_conf=None*)
Create a new session in Livy.

The py_files, files, jars and archives arguments are lists of URLs, e.g. ["s3://bucket/object", "hdfs://path/to/file", ...] and must be reachable by the Spark driver process. If the provided URL has no scheme, it's considered to be relative to the default file system configured in the Livy server.

URLs in the py_files argument are copied to a temporary staging area and inserted into Python's sys.path ahead of the standard library paths. This allows you to import .py, .zip and .egg files in Python.

URLs for jars, py_files, files and archives arguments are all copied to the same working directory on the Spark cluster.

The driver_memory and executor_memory arguments have the same format as JVM memory strings with a size unit suffix ("k", "m", "g" or "t") (e.g. 512m, 2g).

See https://spark.apache.org/docs/latest/configuration.html for more information on Spark configuration properties.

**Parameters**

- **kind** (SessionKind) – The kind of session to create.
- **proxy_user** (Optional[str]) – User to impersonate when starting the session.
- **jars** (Optional[List[str]]) – URLs of jars to be used in this session.
- **py_files** (Optional[List[str]]) – URLs of Python files to be used in this session.
- **files** (Optional[List[str]]) – URLs of files to be used in this session.
- **driver_memory** (Optional[str]) – Amount of memory to use for the driver process (e.g. '512m').
- **driver_cores** (Optional[int]) – Number of cores to use for the driver process.
- **executor_memory** (Optional[str]) – Amount of memory to use per executor process (e.g. '512m').
- **executor_cores** (Optional[int]) – Number of cores to use for each executor.
- **num_executors** (Optional[int]) – Number of executors to launch for this session.

- **archives** (Optional[List[str]]) – URLs of archives to be used in this session.
- **queue** (Optional[str]) – The name of the YARN queue to which submitted.
- **name** (Optional[str]) – The name of this 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