Guide to Using Spark with Third-Party Libraries

# How to Add a Library to PySpark

This guide provides step-by-step instructions on how to install and use a third-party Python library in a PySpark environment running on a distributed Spark cluster. We'll walk through installing the 'requests' library on all worker nodes and the master node, and demonstrate how to use it within a PySpark job.

## Step 1: Log in to Worker Nodes and Install the Library

1. Log in to Worker Node 1:  
docker-compose exec worker1 bash

2. Install the Library using pip3:  
pip3 install requests

3. Exit Worker Node 1:  
exit

4. Log in to Worker Node 2:  
docker-compose exec worker2 bash

5. Install the Library using pip3:  
pip3 install requests

6. Exit Worker Node 2:  
exit

## Step 2: Log in to the Master Node and Install the Library

1. Log in to the Master Node:  
docker-compose exec master bash

2. Install the Library using pip3:  
pip3 install requests

## Step 3: Start PySpark

1. Start PySpark:  
pyspark

## Step 4: Run a PySpark Job that Uses the requests Library

1. Create a Simple DataFrame:

data = [('https://api.github.com',), ('https://httpbin.org/get',)]  
df = spark.createDataFrame(data, ['url'])  
df.show()

2. Use the requests Library to Fetch Data from Each URL:

import requests  
from pyspark.sql.functions import udf  
from pyspark.sql.types import StringType  
  
def fetch\_url(url):  
 try:  
 response = requests.get(url)  
 return f"Status: {response.status\_code}, URL: {url}"  
 except Exception as e:  
 return f"Failed to fetch {url}: {str(e)}"  
  
fetch\_url\_udf = udf(fetch\_url, StringType())  
  
result\_df = df.withColumn('response', fetch\_url\_udf(df['url']))  
result\_df.show(truncate=False)

3. Show the Results:

result\_df.show(truncate=False)