

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



APACHE SPARK

Launch Spark on Google Colab and connect to SparkUI

Start experimenting or learning fast!



Alex Litvinov · [Follow](#)

Published in Level Up Coding · 4 min read · Nov 7, 2023



85





Photo by [Christopher Burns](#) on [Unsplash](#)

Without a doubt, **Apache Spark** is currently the most popular framework for large-scale data processing. It features caching data in memory which makes calculations blazing fast by eliminating the overhead of writing to disk. Additionally, it offers a rich set of APIs in Java, Scala, Python, and R.

On the other hand, who doesn't love Jupyter notebooks? They are highly useful because of their interactive coding environment, support for mixed media elements, and the ability to combine code with documentation and data visualization. Google Colab, in turn, takes the convenience of Jupyter notebooks to the next level by providing a cloud-based platform that offers free GPU access, collaborative features, and the ability to run notebooks without the need for local installations or powerful hardware. Can we harness the full power of Spark within the comfort of Google Colab?

Absolutely! Today we will set up Google Colab to work with Spark with the ability to access SparkUI. It is useful because it allows users to monitor the progress and performance of Spark applications, displaying real-time information on tasks, stages, and resource utilization, helping developers optimize and troubleshoot their Spark jobs efficiently.

The setup we're about to discuss can be useful for data scientists who need to conduct some quick POC. And of course, its primary target audience is people who start to learn Spark and want to dive right into it, bypassing the sometimes cumbersome initial installation and setup as much as possible.

Installation

We will begin by installing `pyspark` itself, along with `findspark` library which is necessary for loading Spark without the need for tedious environment variables setting.



Search

Write



To be able to access SparkUI which resides on Colab's localhost, we need to make it available from the internet. For that, we will forward it using ngrok. Ngrok is a tool that simplifies the process of exposing your local server ports to the Internet by creating a secure tunnel to locally hosted applications using a reverse proxy. This is done by providing a temporary public URL. In our case, local equals Colab.

To use Ngrok, you need to register on their website first. After registering, an automatically generated AuthToken is given which can be used later to authenticate with ngrok.

To interact with ngrok we will install its Python wrapper pyngrok

```
%%capture
!pip install pyspark
```

```
!pip install findspark  
!pip install pyngrok
```

Set up Spark session

`findspark.init()` performs several crucial tasks, including:

- Locating the Spark installation on the system and adding it to the Python PATH.
- Setting the necessary environment variables.

After that we have a `SparkSession` object available, enabling you to configure and submit Spark jobs from your Python code.

```
import findspark  
  
findspark.init()  
  
from pyspark.sql import SparkSession  
  
spark = SparkSession.builder \  
    .appName('testColab') \  
    .getOrCreate()
```

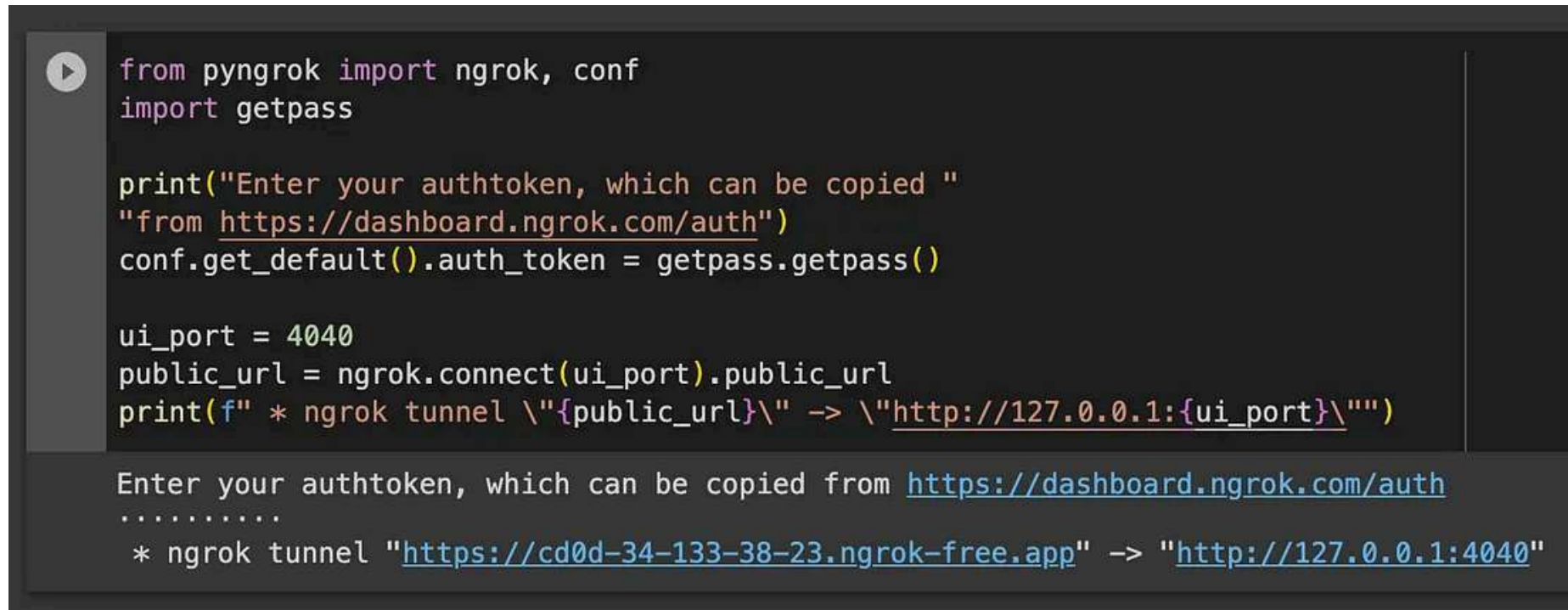
Start a tunnel to access SparkUI

```
from pyngrok import ngrok, conf
import getpass

print("Enter your authtoken, which can be copied "
      "from https://dashboard.ngrok.com/get-started/your-authtoken")
conf.get_default().auth_token = getpass.getpass()

ui_port = 4040
public_url = ngrok.connect(ui_port).public_url
print(f" * ngrok tunnel \"{public_url}\" -> \"http://127.0.0.1:{ui_port}\"")
```

After you enter your AuthToken copied from the ngrok dashboard you're all set — SparkUI will be accessible on the provided temporary public URL

A screenshot of a Google Colab code cell. The code imports pyngrok and getpass, prompts for an auth token, and connects to a public URL. The output shows the temporary URL created by ngrok.

```
from pyngrok import ngrok, conf
import getpass

print("Enter your authtoken, which can be copied "
      "from https://dashboard.ngrok.com/auth")
conf.get_default().auth_token = getpass.getpass()

ui_port = 4040
public_url = ngrok.connect(ui_port).public_url
print(f"* ngrok tunnel \"{public_url}\" -> \"http://127.0.0.1:{ui\_port}\")")
```

Enter your authtoken, which can be copied from <https://dashboard.ngrok.com/auth>
.....
* ngrok tunnel "<https://cd0d-34-133-38-23.ngrok-free.app>" -> "<http://127.0.0.1:4040>"

Google Colab cell showing the temporary URL created by ngrok

For the sake of demonstration let's download a file and count the number of rows in it.

```
from pyspark import SparkFiles

file_url = 'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023'
spark.sparkContext.addFile(file_url)
```

```
df = spark.read.csv(SparkFiles.get('yellow_tripdata_2023-07.parquet'),  
                    header=True)  
  
df.count()
```

Voilà! We can follow the execution on the SparkUI forwarded using ngrok.
Now it's time for you to take over and start running your data pipelines.

Spark 3.5.0 Jobs Stages Storage Environment Executors SQL / DataFrame testColab application UI

Spark Jobs (?)

User: root
Total Uptime: 41 s
Scheduling Mode: FIFO
Completed Jobs: 3

Event Timeline

Completed Jobs (3)

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/10/20 21:10:32	96 ms	1/1 (1 skipped)	1/1 (2 skipped)
1	count at NativeMethodAccessorImpl.java:0 count at NativeMethodAccessorImpl.java:0	2023/10/20 21:10:31	1.0 s	1/1	2/2
0	csv at NativeMethodAccessorImpl.java:0 csv at NativeMethodAccessorImpl.java:0	2023/10/20 21:10:29	1 s	1/1	1/1

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

SparkUI on Google Colab accessed through ngrok

In conclusion, the discussed setup simplifies the installation process, enabling data scientists and learners to harness the power of Spark without the complexities of local installation and setup, all while monitoring performance through SparkUI.

Follow-along Colab Notebook

All the code is available on GitHub:

medium_articles/Spark_in_Colab.ipynb at main · aaalexlit/medium_articles

Contribute to aaalexlit/medium_articles development by creating an account on GitHub.

github.com

Happy experimenting!

References

Demo data: The New York City Taxi and Limousine Commission (2023). TLC Trip Record Data. <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>