

[Go back](#)

Integrating PySpark notebook with S3

Fri 24 January 2020

Introduction

In my post [Using Spark to read from S3](https://www.jitsejan.com/using-spark-to-read-from-s3.html) (<https://www.jitsejan.com/using-spark-to-read-from-s3.html>) I explained how I was able to connect Spark to AWS S3 on a Ubuntu machine. Last week I was trying to connect to S3 again using Spark on my local machine, but I wasn't able to read data from our datalake. Our datalake is hosted in the `eu-west-2` region which apparently requires you to specify the version of authentication. Instead of setting up the right environment on my machine and reconfigure everything, I chose to update the Docker image from my notebook repo (<https://github.com/jitsejan/notebooks>) so I could test locally on my Mac before pushing it to my server. Instead of configuring both my local and remote environment I can simply spin up the Docker container and have two identical environments.

Implementation

Rather than providing the AWS credentials in the Spark config, I want to keep things simple and only have one credentials file from where I will read the important information. The contents of `~/.aws/credentials` specify just one account in this example, but this is where I have specified all the different AWS accounts we are using. This file will be copied to the Docker container by mounting the local `aws` folder inside the Docker instance.

```
[prod]
aws_access_key_id = xxxxxxxxyyyyyyy
aws_secret_access_key = zzzzzzzzyyyyyyy
region = eu-west-2
```

The Dockerfile consists of different steps. I have stripped down the Dockerfile to only install the essentials to get Spark working with S3 and a few extra libraries (like `nltk`) to play with some data. A few things to note:

- The base image is the `pyspark-notebook` provided by Jupyter (<https://github.com/jupyter/docker-stacks/tree/master/pyspark-notebook>).
- Some packages are installed to be able to install the rest of the Python requirements.
- The Jupyter configuration (see below) is copied to the Docker image.
- Two libraries for Spark are downloaded to interact with AWS. These particular versions seem to work well, where newer versions caused different issues during my testing.
- The Python packages are installed defined in the `requirements.txt`.

- The Jupyter packages and extensions are installed and enabled.
- The notebook is started in Jupyter *lab* mode.

```
FROM jupyter/pyspark-notebook
USER root
# Add essential packages
RUN apt-get update && apt-get install -y build-essential curl git gnupg2 nano
apt-transport-https software-properties-common
# Set locale
RUN apt-get update && apt-get install -y locales \
    && echo "en_US.UTF-8 UTF-8" > /etc/locale.gen \
    && locale-gen
# Add config to Jupyter notebook
COPY jupyter/jupyter_notebook_config.py /home/jovyan/.jupyter/
RUN chmod -R 777 /home/jovyan/
# Spark libraries
RUN wget https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk/1.7.4/aws-
java-sdk-1.7.4.jar -P $SPARK_HOME/jars/
RUN wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-
aws/2.7.3/hadoop-aws-2.7.3.jar -P $SPARK_HOME/jars/

USER $NB_USER
# Install Python requirements
COPY requirements.txt /home/jovyan/
RUN pip install -r /home/jovyan/requirements.txt
# Install NLTK
RUN python -c "import nltk; nltk.download('popular')"
# Custom styling
RUN mkdir -p /home/jovyan/.jupyter/custom
COPY custom/custom.css /home/jovyan/.jupyter/custom/
# NB extensions
RUN jupyter contrib nbextension install --user
RUN jupyter nbextensions_configurator enable --user
# Run the notebook
CMD ["/opt/conda/bin/jupyter", "lab", "--allow-root"]
```

The Jupyter configuration file sets up the notebook environment. In my case I set the password, the startup directory and the IP restrictions.

```
""" jupyter_notebook_config.py """
c = get_config()
c.InteractiveShell.ast_node_interactivity = "all"
c.NotebookApp.allow_origin = '*'
c.NotebookApp.ip = '*'
c.NotebookApp.notebook_dir = '/opt/notebooks/'
c.NotebookApp.open_browser = False
c.NotebookApp.password = u'sha1:a123:345345'
c.NotebookApp.port = 8558
```

The `docker-compose.yml` contains the setup of the Docker instance. The most important parts of the compose file are:

- The notebook and data folder are mapped from the Docker instance to a local folder.
- The credential file is mapped from the Docker machine to the local machine.
- The public port is set to 8558.

```
version: '3'
services:
  jitsejan-pyspark:
    user: root
    privileged: true
    image: jitsejan/pyspark-notebook
    restart: always
    volumes:
      - ./notebooks:/opt/notebooks
      - ./data:/opt/data
      - $HOME/.aws/credentials:/home/jovyan/.aws/credentials:ro
    environment:
      - GRANT_SUDO=yes
    ports:
      - "8558:8558"
```

Execution

I am running Docker version 19.03.5 at the time of writing.

```
~/code/notebooks > master $ docker version
Client: Docker Engine - Community
 Version:           19.03.5
 API version:       1.40
 Go version:        go1.12.12
 Git commit:        633a0ea
 Built:             Wed Nov 13 07:22:34 2019
 OS/Arch:           darwin/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:           19.03.5
  API version:       1.40 (minimum version 1.12)
  Go version:        go1.12.12
  Git commit:        633a0ea
  Built:             Wed Nov 13 07:29:19 2019
  OS/Arch:           linux/amd64
  Experimental:      false
 containerd:
  Version:           v1.2.10
  GitCommit:         b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:           1.0.0-rc8+dev
  GitCommit:         3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:           0.18.0
  GitCommit:         fec3683
```

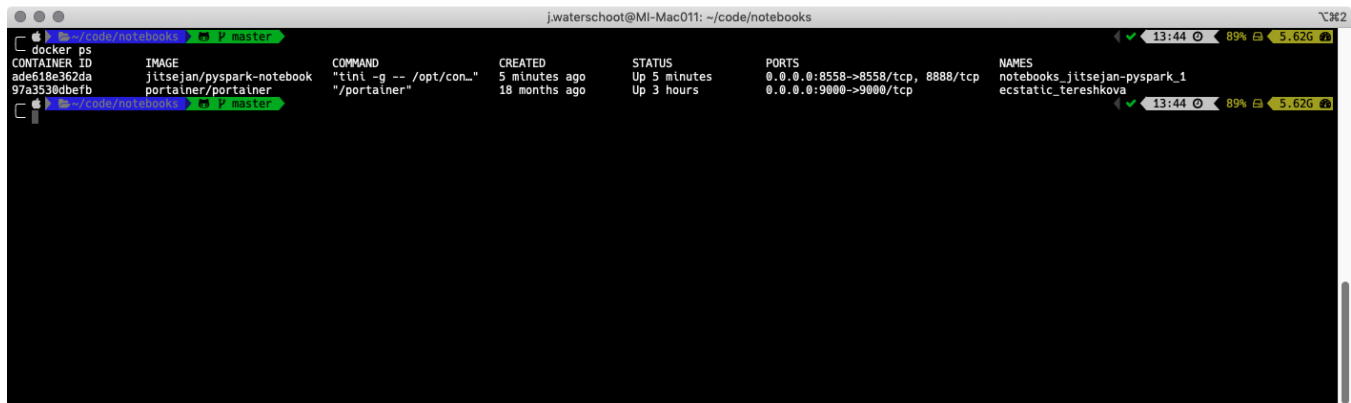
Run `docker-compose up` after creating the compose file to spin up the notebook.

```
~/code/notebooks > master $ docker-compose up
Creating network "notebooks_default" with the default driver
Creating notebooks_jitsejan-pyspark_1 ... done
Attaching to notebooks_jitsejan-pyspark_1
jitsejan-pyspark_1 | [I 13:38:47.211 LabApp] Writing notebook server cookie
secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
jitsejan-pyspark_1 | [W 13:38:47.474 LabApp] WARNING: The notebook server is
listening on all IP addresses and not using encryption. This is not recommended.
jitsejan-pyspark_1 | [I 13:38:47.516 LabApp]
[jupyter_nbextensions_configurator] enabled 0.4.1
jitsejan-pyspark_1 | [I 13:38:48.766 LabApp] JupyterLab extension loaded from
/opt/conda/lib/python3.7/site-packages/jupyterlab
jitsejan-pyspark_1 | [I 13:38:48.767 LabApp] JupyterLab application directory
is /opt/conda/share/jupyter/lab
jitsejan-pyspark_1 | [I 13:38:49.802 LabApp] Serving notebooks from local
directory: /opt/notebooks
jitsejan-pyspark_1 | [I 13:38:49.802 LabApp] The Jupyter Notebook is running
at:
jitsejan-pyspark_1 | [I 13:38:49.802 LabApp] http://ade618e362da:8558/
jitsejan-pyspark_1 | [I 13:38:49.803 LabApp] Use Control-C to stop this server
and shut down all kernels (twice to skip confirmation).
```

The container is now running on port 8558 :

```
~/code/notebooks > master $ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
ade618e362da	jitsejan/pyspark-notebook	"tini -g -- /opt/con..."	2 minutes ago
notebooks_jitsejan-pyspark_1	portainer/portainer	0.0.0.0:8558->8558/tcp, 8888/tcp	Up 2 minutes



For convenience I am running Portainer (<https://www.portainer.io/>) because it is easier to get an overview of the containers running in Docker instead of using the CLI.. It also helps to clean up all the orphan images.

The screenshot shows the Portainer web interface. On the left is a dark sidebar with navigation links: LOCAL (Dashboard, App Templates, Stacks, Containers, Images, Networks, Volumes, Events, Engine), PORTAINER SETTINGS (User management, Endpoints, Registries, Settings), and the Portainer.io logo at the bottom. The main panel displays 'Container details' for a container named 'notebooks_jitsejan-pyspark_1'. It includes an 'Actions' bar with buttons for Start, Stop, Kill, Restart, Pause, Resume, Remove, Recreate, and Duplicate/Edit. Below is a 'Container status' table with fields: ID (ade618e362da8417efd327bbe5329cd745196bab3479ea2ebc6f20d4ecb426e), Name (notebooks_jitsejan-pyspark_1), Status (Running for 6 minutes), Created (2020-01-24 13:38:44), and Start time (2020-01-24 13:38:45). There are links for Stats, Logs, Console, and Inspect. The 'Access control' section shows 'Ownership' as 'public' with a 'Change ownership' link. The 'Create image' section explains how to create an image from the container and includes a form with 'Name' (e.g., myImage:myTag), 'Registry' (DockerHub), and a 'Create' button. A note states: 'Note: if you don't specify the tag in the image name, latest will be used.'

The screenshot shows a web browser with two tabs: 'Portainer' and 'Jupyter Notebook'. The address bar shows 'localhost:8558/login?next=%2Ftab%3F'. The page features the Jupyter logo and a login form with a 'Password:' label, a text input field, and a 'Log in' button.

Code

To connect Spark to S3 I will use the `credentials` file as configuration file and use the `configparser` library to read the parameters.

```
from configparser import ConfigParser

config_object = ConfigParser()
config_object.read("/home/jovyan/.aws/credentials")
profile_info = config_object["prod"]
```

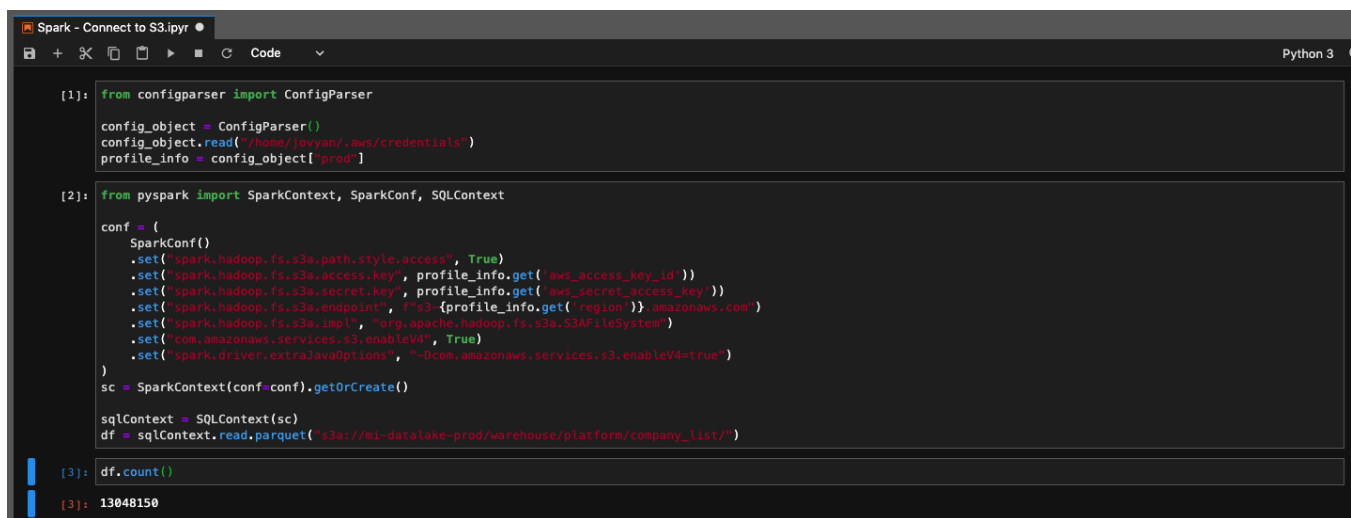
Since the configuration now contains the production account data I can use it so set the parameters for the Spark context. Note that all these parameters are required to connect to the data on S3. The access key and secret are always mentioned in tutorials, but it took me a while to figure out I need to specify the `endpoint` and enable V4.

```
from pyspark import SparkContext, SparkConf, SQLContext

conf = (
    SparkConf()
    .set("spark.hadoop.fs.s3a.path.style.access", True)
    .set("spark.hadoop.fs.s3a.access.key",
profile_info.get('aws_access_key_id'))
    .set("spark.hadoop.fs.s3a.secret.key",
profile_info.get('aws_secret_access_key'))
    .set("spark.hadoop.fs.s3a.endpoint", f"s3-
{profile_info.get('region')}.amazonaws.com")
    .set("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem")
    .set("com.amazonaws.services.s3.enableV4", True)
    .set("spark.driver.extraJavaOptions", "-
Dcom.amazonaws.services.s3.enableV4=true")
)
```

With the above configuration I initialize the Spark context and can read from our datalake.

```
sc = SparkContext(conf=conf).getOrCreate()
sqlContext = SQLContext(sc)
df = sqlContext.read.parquet("s3a://mi-datalake-
prod/warehouse/platform/company_list/")
```



```
[1]: from configparser import ConfigParser

config_object = ConfigParser()
config_object.read("/home/jovyan/.aws/credentials")
profile_info = config_object["prod"]

[2]: from pyspark import SparkContext, SparkConf, SQLContext

conf = (
    SparkConf()
    .set("spark.hadoop.fs.s3a.path.style.access", True)
    .set("spark.hadoop.fs.s3a.access.key", profile_info.get("aws_access_key_id"))
    .set("spark.hadoop.fs.s3a.secret.key", profile_info.get("aws_secret_access_key"))
    .set("spark.hadoop.fs.s3a.endpoint", f"s3-{profile_info.get('region')}.amazonaws.com")
    .set("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem")
    .set("com.amazonaws.services.s3.enableV4", True)
    .set("spark.driver.extraJavaOptions", "-Dcom.amazonaws.services.s3.enableV4=true")
)
sc = SparkContext(conf=conf).getOrCreate()

sqlContext = SQLContext(sc)
df = sqlContext.read.parquet("s3a://ml-datalake-prod/warehouse/platform/company_list/")

[3]: df.count()

[3]: 13048150
```

Hope this helps!

[Python \(./tag/python.html\)](#)

[PySpark \(./tag/pyspark.html\)](#)

[S3 \(./tag/s3.html\)](#)

[dataframe \(./tag/dataframe.html\)](#)

[Spark \(./tag/spark.html\)](#)

[Docker \(./tag/docker.html\)](#)

Go back

© JJ's World (.) | Powered by Pelican (<https://getpelican.com/>) | Hosted on Cloudflare Pages (<https://pages.cloudflare.com/>) | 2008 - 2021