Hands-on Tour of Apache Spark in 5 Minutes

with Python

If you have any errors in completing this tutorial. Please ask questions or notify us on Hortonworks Community Connection!

Introduction

Apache Spark is a fast, in-memory data processing engine with elegant and expressive development APIs in Scala, Java, Python, and R that allow data workers to efficiently execute machine learning algorithms that require fast iterative access to datasets (see Spark API Documentation for more info). Spark on Apache Hadoop YARN enables deep integration with Hadoop and other YARN enabled workloads in the enterprise.

In this tutorial, we will introduce the basic concepts of Apache Spark and the first few necessary steps to get started with Spark using an Apache Zeppelin Notebook on a Hortonworks Data Platform (HDP) Sandbox.

Prerequisites

This tutorial is a part of series of hands-on tutorials to get you started with HDP using Hortonworks sandbox. Please ensure you complete the prerequisites before proceeding with this tutorial.

- Downloaded and Installed latest Hortonworks Sandbox
- Learning the Ropes of the Hortonworks Sandbox

Concepts

At the core of Spark is the notion of a **Resilient Distributed Dataset** (RDD), which is an immutable collection of objects that is partitioned and distributed across multiple physical nodes of a YARN cluster and that can be operated in parallel.

Typically, RDDs are instantiated by loading data from a shared filesystem, HDFS, HBase, or any data source offering a Hadoop InputFormat on a YARN cluster.

Once an RDD is instantiated, you can apply a series of operations . All operations fall into one of two types: transformations or actions . **Transformation** operations, as the name suggests, create new datasets from an existing RDD and build out the processing Directed Acyclic Graph (DAG) that can then be applied on the partitioned dataset across the YARN cluster. An **Action** operation, on the other hand, executes DAG and returns a value.

Let's try it out.

A Hands-On Example

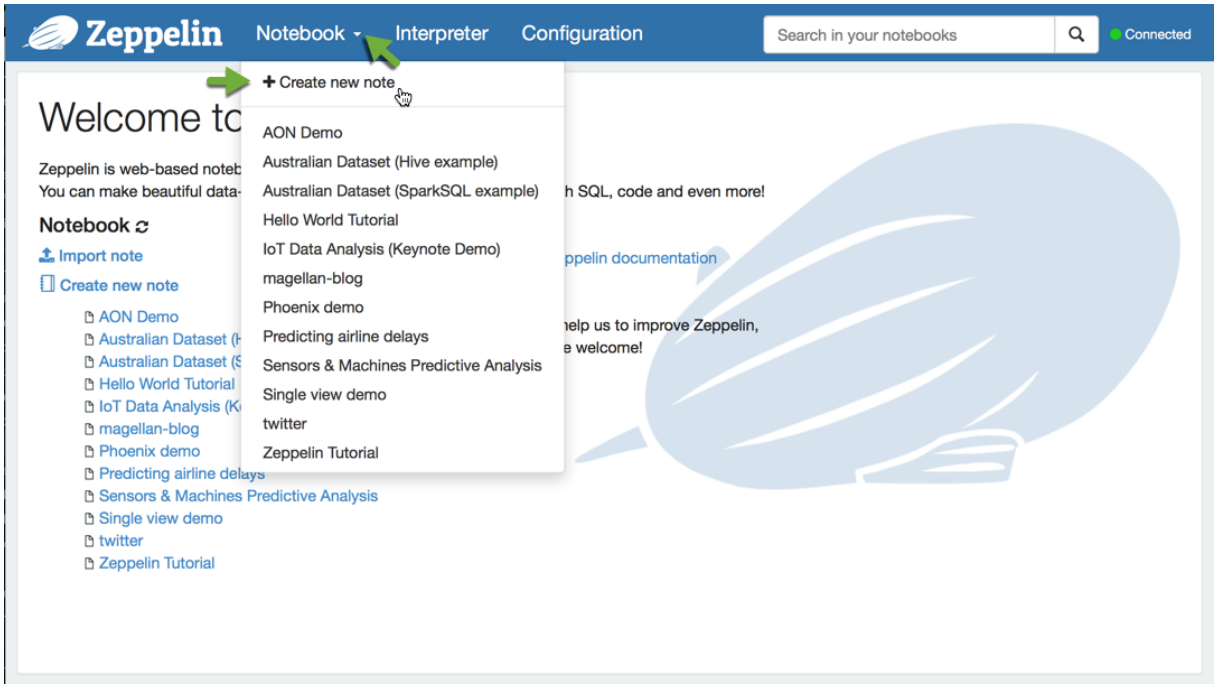
First, let's launch **Zeppelin** from your browser.

Go to `http://<host IP>:9995`.

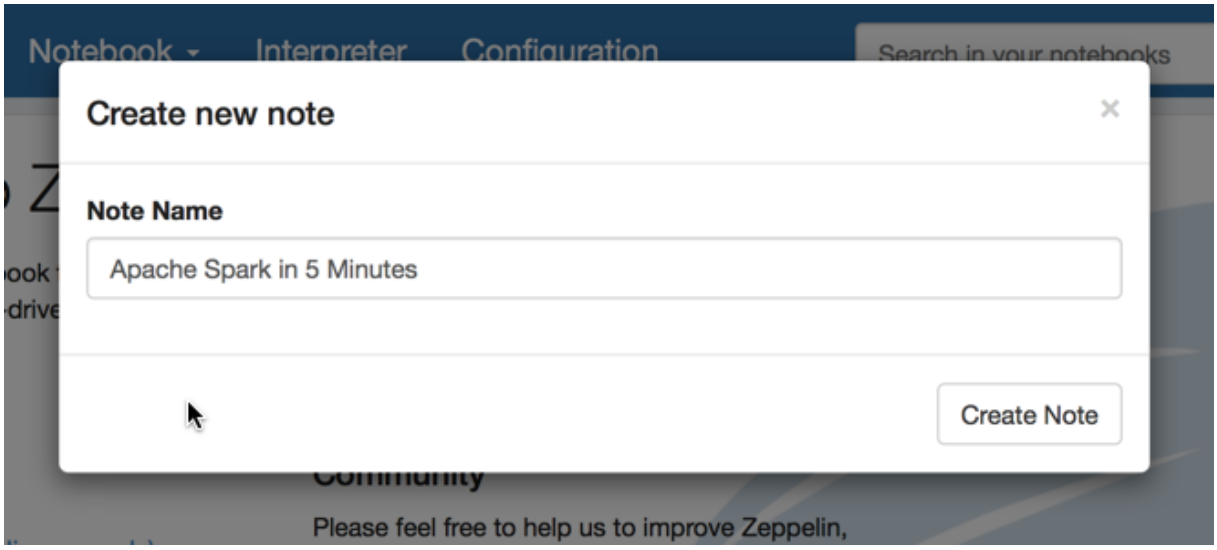
For example, the default local **VirtualBox** address is http://127.0.0.1:9995 and the default local **VmWare** address is http://172.16.148.128:9995 .

Note: In local mode your `host IP` should be `127.0.0.1` for VirtualBox and `172.16.148.128` for VmWare, however if you are running your Sandbox in the cloud, review Learning the Ropes to learn how to determine your `host IP` address.

Next, select **Create new note** from **Notebook** dropdown menu:



Give your notebook a name. I named my notebook *Apache Spark in 5 Minutes*



Zeppelin comes with several interpreters pre-configured on Sandbox. In this tutorial we will use a shell interpreter

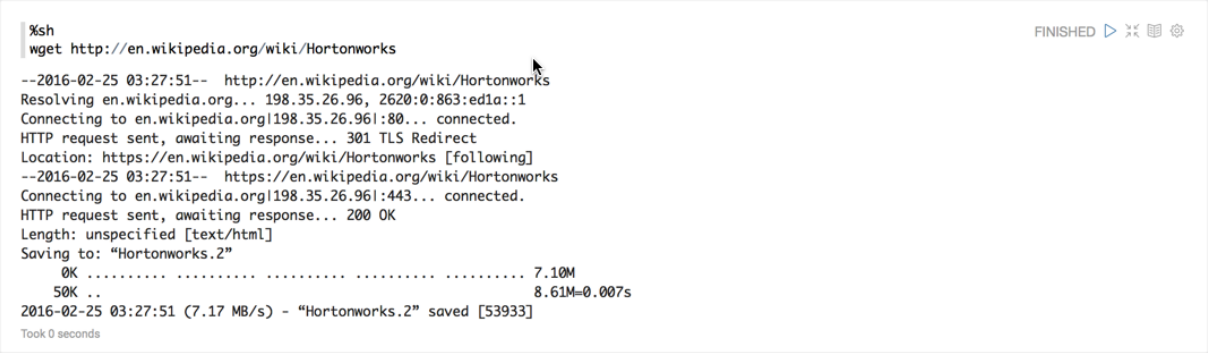
`%sh` and a pyspark interpreter `%pyspark`.

Let's start with a shell interpreter `%sh` and bring in some Hortonworks related Wikipedia data.

Type the following in your Zeppelin Notebook and hit **shift + enter** to execute the code:

```
%sh
wget http://en.wikipedia.org/wiki/Hortonworks
```

You should see an output similar to this



Next, let's copy the data over to HDFS. Type and execute the following:

```
%sh
hadoop fs -put ~/Hortonworks /tmp
```

Now we are ready to run a simple Python program with Spark. This time we will use the python interpreter `%pyspark`. Copy and execute this code:

```
%pyspark
myLines = sc.textFile('hdfs://sandbox.hortonworks.com/tmp/Hortonworks')
myLinesFiltered = myLines.filter( lambda x: len(x) > 0 )
count = myLinesFiltered.count()
print count
```

When you execute the above you should get only a number as an output. I got `311` but it may vary depending on the Wikipedia entry.

Let's go over what's actually going on. After the python interpreter `%pyspark` is initialized we instantiate an RDD using a Spark Context `sc` with a `Hortonworks` file on HDFS:

```
myLines = sc.textFile('hdfs://sandbox.hortonworks.com/tmp/Hortonworks')
```

After we instantiate the RDD, we apply a transformation operation on the RDD. In this case, a simple transformation operation using a Python lambda expression to filter out all the empty lines:

```
myLinesFiltered = myLines.filter( lambda x: len(x) > 0 )
```

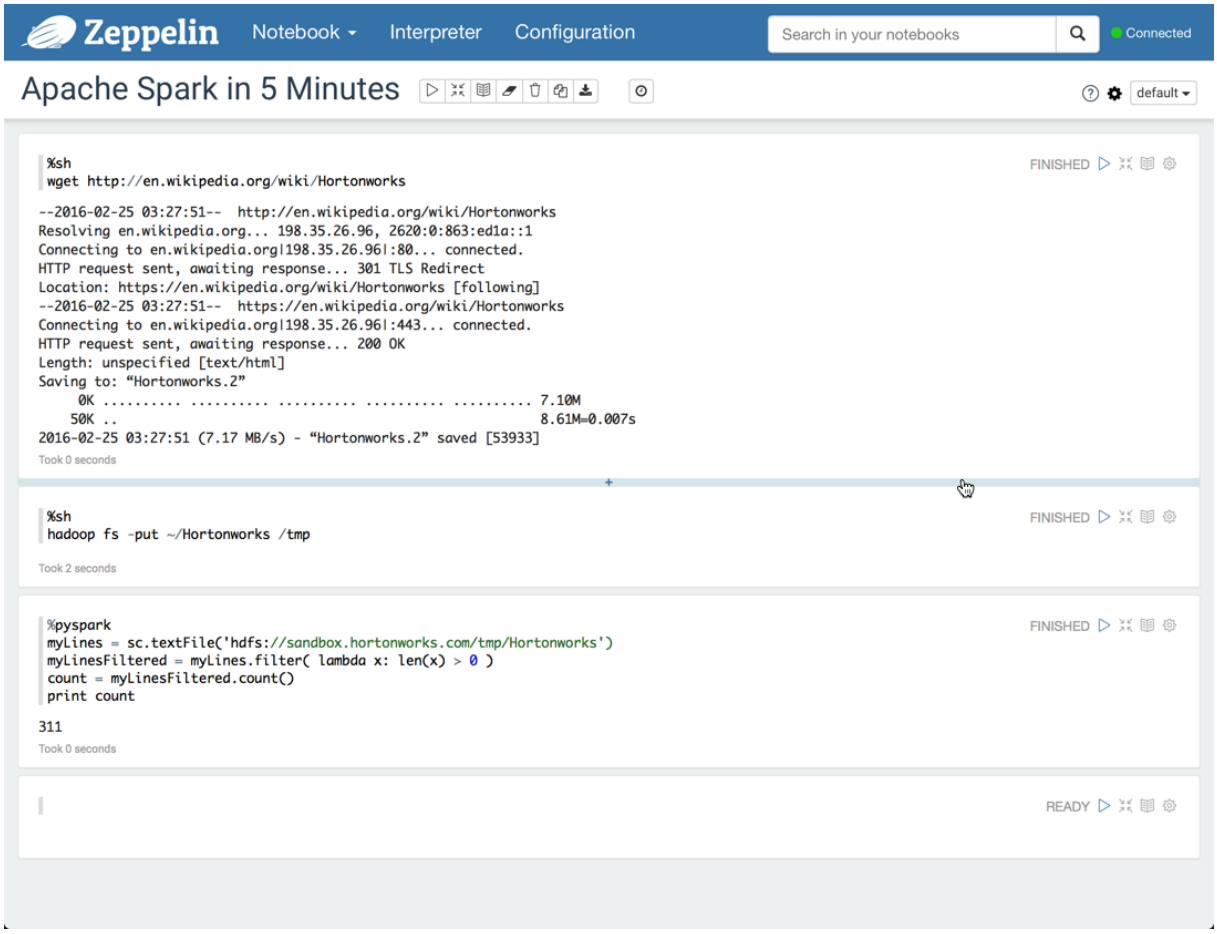
At this point, the transformation operation above did not touch the data in any way. It has only modified the processing graph.

We finally execute an action operation using the aggregate function `count()`, which then executes all the transformation operations:

```
count = myLinesFiltered.count()
```

Lastly, with `print count` we display the final count value, which returns `311`.

That's it! Your complete notebook should look like this after you run your code in all paragraphs:



We hope that this little example wets your appetite for more ambitious data science projects on the Hortonworks Data Platform (HDP) Sandbox.

If you're feeling adventurous checkout our other great Apache Spark & Hadoop tutorials.

Tutorial Q&A and Reporting Issues

If you need help or have questions with this tutorial, please first check HCC for existing Answers to questions on this tutorial using the Find Answers button. If you don't find your answer you can post a new HCC question for this tutorial using the Ask Questions button.

Find Answers **Ask Questions**

Tutorial Name: **A Hands-on Tour of Apache Spark**
HCC Tutorial Tag: **tutorial-360** and **hdp-2.4.0**

All Hortonworks, partner and community tutorials are posted in the Hortonworks github and can be contributed via the Hortonworks Tutorial Collaboration Guide. If you are certain there is an issue or bug with the tutorial, please create an issue on the repository and we will do our best to resolve it!

+1 408 675-0983

+1 855 8-HORTON

INTL: +44 (0) 20 3826 1405