Creating Zeppelin Runbooks

Date of Publish: 2018-10-15

http://docs.hortonworks.com

Contents

Creating Runbooks Using Apache Zeppelin

Apache Zeppelin is a web-based notebook that supports interactive data exploration, visualization, sharing and collaboration.

HCP users will use Zeppelin at two levels:

- Senior analysts and data scientists can use Zeppelin to produce notebooks to analyze data and to create recreatable investigations or runbooks for junior analysts.
- Junior analysts can use recreatable investigations or runbooks in Zeppelin to discover cybersecurity issues much like they do with the Metron Dashboard. However, Zeppelin can handle larger groups of data.

Setting up Zeppelin to Run with HCP

You can import the Zeppelin Notebook using Ambari or manually. To complete your set up you'll need to use Zeppelin interpretors and load the telemetry information.

Procedure

- Setting up Zeppelin is very simple. To access Zeppelin, go to http://$ZEPPELIN_HOST:9995.
- In addition to this documentation, there are three other sources for Zeppelin information.
  - The Zeppelin installation for HCP provides a couple sample notes including tutorials specific to HCP. These notes are listed on the left side of the Welcome screen and in the Notebook menu.
  - Hortonworks Zeppelin documentation provides information on launching and using Zeppelin.
  - Apache Zeppelin documentation provides information on Zeppelin basic features, supported interpreters, and more.

Using Zeppelin Interpreters

When you install Zeppelin on HCP the installation includes the interpreter for Spark. Spark is the main backend processing engine for Zeppelin. Spark is also a front end for Python, Scala, and SQL and you can use any of these languages to analyze the telemetry data.

Loading Telemetry Information into Zeppelin

Before you can analyze telemetry information in Zeppelin, you must first download it from Metron. Metron archives the fully parsed, enriched, and triaged telemetry for each sensor in HDFS. This archived telemetry information is simply raw JSON files which makes it simple to parse and analyze the information with Zeppelin.

The following is an example of some Bro telemetry information.

```
%sh

hdfs dfs -ls -C -R /apps/metron/indexing/indexed/bro
/apps/metron/indexing/indexed/bro/enrichment-null-0-0-1484124296101.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-0-1484128332104.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-0-1484131460758.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-1-1484217861096.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-10-1484995461039.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-11-1485081861043.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-12-1485168261040.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-13-1485254661040.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-14-1485341061047.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-15-1485427461040.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-16-1485513861039.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-17-1485600261045.json
```

```
/apps/metron/indexing/indexed/bro/enrichment-null-0-18-1485686661035.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-19-1485773061037.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-2-1484304261042.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-20-1485859461037.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-21-1485945861039.json
/apps/metron/indexing/indexed/bro/enrichment-null-0-22-1486032261036.json
```

You can use Spark to load the archived information from HDFS into Zeppelin.

For example if you are loading information received from Bro, your command would like the following:

```
%spark
sqlContext.read.json("hdfs:///apps/metron/indexing/indexed/
bro").cache().registerTempTable("bro")
```

Working with Zeppelin

The Zeppelin user interface consists of notes that are divided into paragraphs. Each paragraph consists of two sections: the code section where you put your source code and the result section where you can see the result of the code execution. When you use the Spark interpreter, you can enter source code in Python, Scala, or SQL. When you run the code from the browser, Zeppelin sends the code to a backend processor such as Spark. The processor or service then returns results; you can then use Zeppelin to review and visualize results in the browser.

For more information on using notes, see Working with Notes.

Using Zeppelin to Create Runbooks

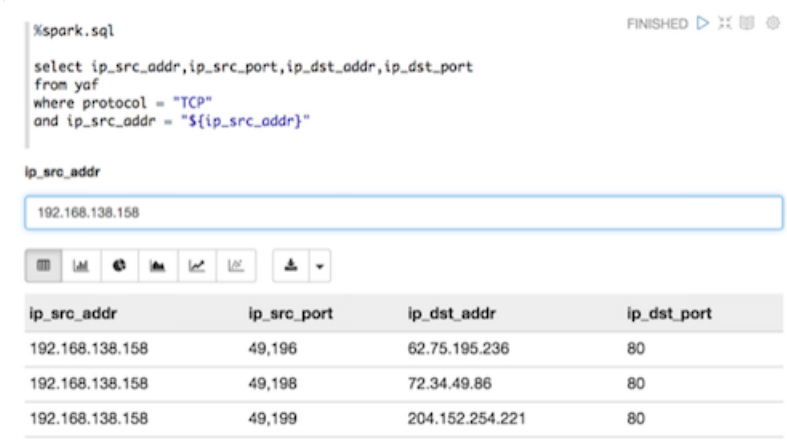
Zeppelin enables data scientists and senior analysts to create workbooks for junior analysts that can be used as runbooks for recreatable investigations. These runbooks can be static, which require no input from the junior analyst, or dynamic, which require the junior analyst to enter or choose information. You can see an example of a static type of notebook in the Metron - YAF Telemetry note. This section provides instructions for creating both kinds of runbooks.

Procedure

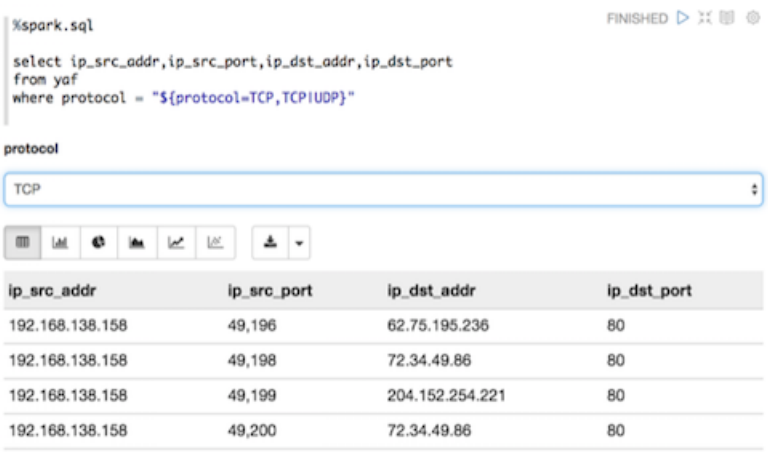
1. Click **Create new note** on the welcome page, or click the **Notebook** menu and choose **+ Create new note**.
2. Type your commands into the blank paragraph in the new note.

   To make your runbook dynamic, use one or more of the dynamic forms that Zeppelin supports:

   - Text input form



   - Text input form with default value
   - Select form



- Checkbox form

When you create a note, it appears in the list of notes on the left side of the home page and in the **Notebook** menu. By default, Zeppelin stores notes in the $ZEPPELIN_HOME/notebook folder.

3. Run your new code by clicking the triangle button in the cell that contains your code.

   Zeppelin attempts to run the code and displays the status near the triangle button: PENDING, RUNNING, ERROR, or FINISHED. Zeppelin also displays another empty paragraph so you can add another command.
4. Continue adding commands until you've completed the runbook.
5. Choose the appropriate type of visualization for your code results from the settings toolbar below the code section of the paragraph.

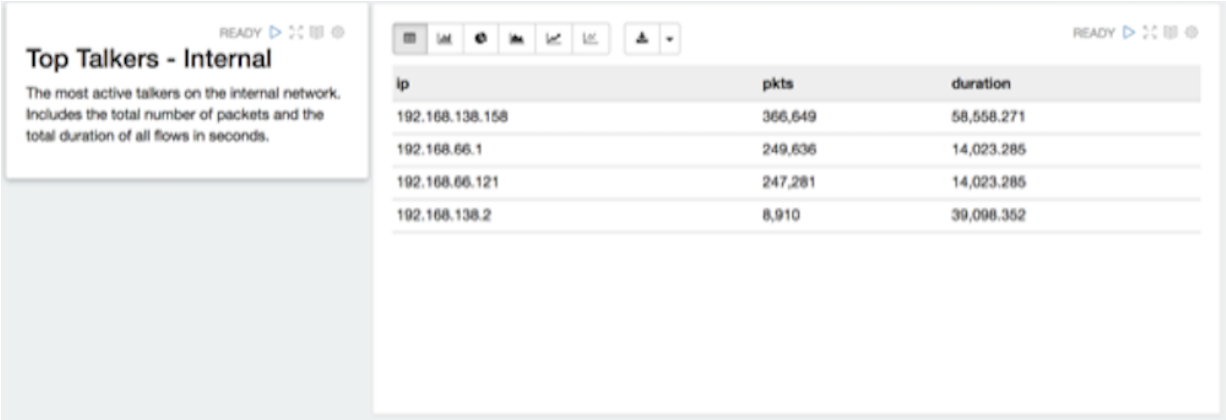
6. If appropriate, notify the junior analyst about the runbook that he can clone and use.

What to do next

The following examples provide sample paragraphs you might want to include in a runbook:

- Top Talkers - Internal

  This paragraph is static and requires no input from the user.



- Flows by hour

  This paragraph is static and requires no input from the user.