

## Setting Up Zeppelin to Run With HCP

Date of publish: 2017-11-06



## Legal Notice

© Cloudera Inc. 2019. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 ("ASLv2"), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER'S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

**Introduction to Using Zeppelin With CCP.....4**

**Setting up Zeppelin to Run with CCP..... 4**

    Using Zeppelin Interpreters.....4

    Loading Telemetry Information into Zeppelin.....6

## Introduction to Using Zeppelin With CCP

The Zeppelin dashboard is intended for use by Security Operations Center (SOC) analysts and investigators.

Like the Metron dashboard, the Zeppelin dashboard can be used to view and analyze the enriched telemetry data provided by CCP. However Zeppelin can be used by a data scientist to create runbooks for recreatable investigations. These runbooks can be static, which require no input, or dynamic, which require you to enter or choose information.

## Setting up Zeppelin to Run with CCP

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

Setting up Zeppelin is very simple. To access Zeppelin, go to `http://$ZEPPELIN_HOST:9995`.

In addition to this documentation, there are two other sources for Zeppelin information.

- The Zeppelin installation for CCP provides a couple sample notes including tutorials specific to Metron. These notes are listed on the left side of the **Welcome** screen and in the **Notebook** menu.
- Zeppelin documentation provides additional information on using Zeppelin.

## Using Zeppelin Interpreters

This section describes how to use Apache Zeppelin interpreters.

Before using an interpreter, ensure that the interpreter is available for use in your note:

1. Navigate to your note.
2. Click on "interpreter binding":



3. Under "Settings", make sure that the interpreter you want to use is selected (in blue text). Unselected interpreters appear in white text:

## Settings

### Interpreter binding

Bind interpreter for this note. Click to Bind/Unbind interpreter. Drag and drop to re-order. The first interpreter on the list becomes default. To create/remove interpreters, go to the Settings page.

The screenshot shows the 'Interpreter binding' section of the Zeppelin Settings page. It contains a list of interpreters, each with a toggle icon (a circular arrow) and a label. The 'spark' interpreter is selected and highlighted in blue. The other interpreters are 'md', 'angular', 'sh', and 'jdbc'. Below the list are 'Save' and 'Cancel' buttons.

Interpreter	Label	Selected
spark	%spark (default), %sql, %dep, %pyspark, %r	Yes
md	%md	No
angular	%angular	No
sh	%sh	No
jdbc	%jdbc	No

Buttons: Save, Cancel

- To select an interpreter, click on the interpreter name to select the interpreter. Each click operates as a toggle.
- You should unselect interpreters that will not be used. This makes your choices clearer. For example, if you plan to use %livy to access Spark, unselect the %spark interpreter.

Whenever one or more interpreters could be used to access the same underlying service, you can specify the precedence of interpreters within a note:

- Drag and drop interpreters into the desired positions in the list.
- When finished, click "Save".

Use an interpreter in a paragraph

To use an interpreter, specify the interpreter directive at the beginning of a paragraph, using the format %[INTERPRETER\_NAME]. The directive must appear before any code that uses the interpreter.

The following paragraph uses the %sh interpreter to access the system shell and list the current working directory:

```
%sh
```

```
pwd
home/zeppelin
```

Some interpreters support more than one form of the directive. For example, the %livy interpreter supports directives for PySpark, PySpark3, SparkR, Spark SQL.

To view interpreter directives and settings, navigate to the Interpreter page and scroll through the list of interpreters or search for the interpreter name. Directives are listed immediately after the name of the interpreter, followed by options and property settings. For example, the JDBC interpreter supports the %jdbc directive:

**jdbc** %jdbc ●

### Option

The interpreter will be instantiated Globally ▾ in shared ▾ process.

☐ Connect to existing process

☐ Set permission

Note: The Interpreter page is subject to access control settings. If the Interpreters page does not list settings, check with your system administrator for more information.

Use interpreter groups

Each interpreter belongs to an interpreter group. Interpreters in the same group can reference each other. For example, if the Spark SQL interpreter and the Spark interpreter are in the same group, the Spark SQL interpreter can reference the Spark interpreter to access its SparkContext.

## Loading Telemetry Information into Zeppelin

Before you can analyze telemetry information in Zeppelin, you must first download it from Cloudera Cybersecurity Platform (CCP).

CCP 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 look like the following:

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