

# Push Data to Splunk from Databricks

---

User Guide 1.0.0

**Note:**

This document is produced by Databricks as a reference. Databricks makes no warranties or guarantees. Information contained within may not be the most up-to-date available.

Information in this document is subject to change without notice. Databricks shall not be liable for any damages resulting from technical errors or omissions that may be present in this document, or from use of this document.

Databricks and the Databricks logo are trademarks or service marks of Databricks, Inc. in the US and other countries. Microsoft Azure and Azure are trademarks of Microsoft Inc. Azure Databricks is a product provided by Microsoft, please see their website for additional information. All other trademarks within this document are property of their respective companies. Copyright 2020 Databricks, Inc. For more information, visit <http://www.databricks.com>.

Technical documentation and support materials include details based on the full set of capabilities and features of a specific release. Access to some functionality requires specific license types (tiers).

## Version Control

#	Document Version	Date	Owner	Document Status	Comments
1	1.0.0	23rd Dec 2020	Crest Data Systems	Draft	Initial Draft

<b>Overview</b>	<b>4</b>
<b>Push Data To Splunk</b>	<b>4</b>
Configure HEC in Splunk Enterprise	4
Enable HEC	4
Create HEC token	5
Configure Sourcetype for the events	6
Steps to configure Sourcetype	6
Using Databricks secret scope to store the Splunk HEC token securely	7
Notebook Parameters	8
Import Splunk Instance's Certificate in Databricks	9
<b>Limitations</b>	<b>10</b>
<b>References</b>	<b>11</b>

# Overview

This document provides information on how to get your data from Databricks into Splunk using a Databricks notebook. The notebook is used to execute queries on a table in Databricks and push the result obtained to Splunk using Splunk HTTP Event Collector (HEC) .

Splunk HEC lets you send data and application events to a Splunk deployment over the HTTP and Secure HTTP (HTTPS) protocols. HEC uses a token-based authentication model. You can generate a token and then configure a logging library or HTTP client with the token to send data to HEC in a specific format. This process eliminates the need for a Splunk forwarder when you send application events. After you enable HEC, as a developer, you can use HEC tokens in your app to send data to HEC. You do not need to include Splunk credentials in your app or supported files.

## Push Data To Splunk

Follow the steps below to push data from Databricks table to Splunk.

### Configure HEC in Splunk Enterprise

Follow the below steps to configure HTTP Event Collector.

#### Enable HEC

- Open Splunk UI.
- Click Settings > Data Inputs.
- Click HTTP Event Collector.

- Click Global Settings.

The screenshot shows the 'Edit Global Settings' window. The 'All Tokens' toggle is highlighted with a blue border and is currently set to 'Enabled'. Below it are several configuration options: 'Default Source Type' with a dropdown menu showing '-- Select Source Type --', 'Default Index' with a dropdown showing 'Default', 'Default Output Group' with a dropdown showing 'None', 'Use Deployment Server' with an unchecked checkbox, 'Enable SSL' with a checked checkbox, and 'HTTP Port Number' with a text input field containing '8088'. At the bottom, there are 'Cancel' and 'Save' buttons.

- In the All Tokens toggle button, select Enabled.
- (Optional) Choose a Default Source Type for all HEC tokens. You can also type in the name of the source type in the text field above the drop-down before choosing the source type.
- (Optional) Choose a Default Index for all HEC tokens.
- (Optional) Choose a Default Output Group for all HEC tokens.
- (Optional) To use a deployment server to handle configurations for HEC tokens, click the Use Deployment Server check box.
- (Optional) To have HEC listen and communicate over HTTPS rather than HTTP, click the Enable SSL checkbox.
- (Optional) Enter a number in the HTTP Port Number field for HEC to listen on. Note: Confirm that no firewall blocks the port number that you specified in the HTTP Port Number field, either on the clients or the Splunk instance that hosts HEC.
- Click Save.

## Create HEC token

- Click Settings > Add Data.
- Click monitor.
- Click HTTP Event Collector.
- In the Name field, enter a name for the token.
- (Optional) In the Source name override field, enter a source name for events that this input generates.
- (Optional) In the Description field, enter a description for the input.

- (Optional) In the Output Group field, select an existing forwarder output group.
- (Optional) If you want to enable indexer acknowledgment for this token, click the Enable indexer acknowledgment checkbox.
- Click Next.
- (Optional) Confirm the source type and the index for HEC events.
- Click Review.
- Confirm that all settings for the endpoint are what you want.
- If all settings are what you want, click Submit. Otherwise, click < to make changes.
- Note the token value that Splunk Web displays for reference.

Reference:

<https://docs.splunk.com/Documentation/Splunk/8.1.1/Data/UsetheHTTPEventCollector>

## Configure Sourcetype for the events

The sourcetype to be associated with the events pushed from this notebook should be configured to have the following properties specified. Note that a user may be required to change the below configurations, based on the data that is being pushed to Splunk.

- LINE\_BREAKER = ([\r\n]+)
- SHOULD\_LINEMERGE = false
- KV\_MODE = json

## Steps to configure Sourcetype

- In Splunk web UI go to settings > Data > Source types
- Select your sourcetype to edit or Click New Sourcetype in the top right-hand corner to create a new sourcetype. In case of a new sourcetype, provide a name for the sourcetype and select the destination app as “search & Reporting” or “Databricks-Addon for Splunk” in the dialog box that opens.
- In the Event Breaks section select Every Line
- In the Advanced section click New setting and add KV\_MODE = JSON as follows.

KV_MODE	json
---------	------

**Note:** In case you are editing an existing sourcetype, make sure that the changes in the Event Breaks and Advanced section do not affect the events that are already associated with this sourcetype.

- Click Save.

If the user wants to change the default timestamp extraction from these events, the user can follow the steps specified here [Configure timestamp recognition](#)

## Using Databricks secret scope to store the Splunk HEC token securely

In the notebook to push data to Splunk, the Splunk HEC token value needs to be specified. Since the HEC token value is sensitive information, the user can configure Databricks secrets to store the token value securely. This enables the user to store the token value securely and not expose it in the notebook in plain text form.

To securely store the Splunk HEC Token value, follow the following steps :

1. Install Databricks CLI and setup authentication using the below commands

**Install:** `pip install databricks-cli`

**Set-up:** `databricks configure --token`

Running the setup command opens an interactive cli. Specify the Databricks host address and the token value prompted and Enter. The token here refers to personal access token in Databricks.Steps to create the token

..<https://docs.databricks.com/dev-tools/api/latest/authentication.html>

Refer: <https://docs.databricks.com/dev-tools/cli/index.html#set-up-the-cli>

2. Create a Databricks backed Secret scope as follows:

`databricks secrets create-scope --scope <scope-name>`

**Or**

`databricks secrets create-scope --scope <scope-name>`

`--initial-manage-principal users`

Refer for details

..<https://docs.databricks.com/security/secrets/secret-scopes.html#create-a-databricks-backed-secret-scope>

3. Create a key and store the secret as follows:

`databricks secrets put --scope <scope-name> --key <key-name>`

The command opens an editor. Enter or paste the Splunk HEC token here and save it.

Refer: <https://docs.databricks.com/security/secrets/secrets.html#create-a-secret>

## Notebook Parameters

The notebook `push_to_splunk` is used to execute queries on a table in Databricks and push the result obtained to Splunk.

The notebook contains the following parameters ( \* denotes mandatory field). Update these parameters based on your Splunk deployment.

- **Protocol\***: The protocol on which Splunk HTTP Event Collector(HEC) runs. Splunk HEC runs on `https` if Enable SSL checkbox is selected while configuring Splunk HEC Token in Splunk, else it runs on `http` protocol. If you do not have access to the Splunk HEC Configuration page, you can ask your Splunk Admin if the `Enable SSL checkbox` is selected or not.
- **Verify Certificate\***: Specify if SSL server certificate verification is required for communication with Splunk. If Splunk HEC runs on http, SSL certificate verification doesn't work. If you set Verify Certificate as True, you may have to import a custom certificate from the Splunk server into Databricks. For this refer section: [Import Splunk Instance's Certificate in Databricks](#). You can get the custom certificate from Splunk Admin.
- **Splunk Ip/Hostname\***: The Splunk IP/Hostname to push data to.
- **Splunk HEC Port\***: The Splunk HEC port. Default Splunk HEC Port is `8088`. You can get this value from Splunk Admin or Splunk HEC Configuration page.
- **Databricks Secret Scope (Either Databricks Secret Scope and Secret Key or Splunk HEC Token are Mandatory Fields)**: The Databricks Secret Scope created using Databricks CLI to store the Secret Key containing the Splunk HEC Token value following step 2 here: [Databricks secret scope](#)
- **Secret Key (Either Databricks Secret Scope and Secret Key or Splunk HEC Token are Mandatory Fields)**: The secret key associated with specified Databricks Secret Scope which securely stores the Splunk HEC Token value following step 3 here: [Databricks secret scope](#)
- **Splunk HEC Token (Either Databricks Secret Scope and Secret Key or Splunk HEC Token are Mandatory Fields)**: The Splunk HEC token value configured in Splunk. It is used when the Databricks Secret Scope and Secret Key are not specified. You can get the value from Splunk Admin or configure as follows: [Configuring Splunk HEC](#)
- **Index**: The Splunk index to push data into. Ensure that the Splunk index specified here is one of the allowed indexes associated with the Splunk HEC Token you are using here. You can get the list of such indexes from the Splunk HEC token Configuration page or from your Splunk Admin.
- **Source**: It indicates the source of an event(in Splunk), that is, where the event originated.



- **Sourcetype:** The sourcetype for an event is used to specify the data structure of an event. Ensure that this sourcetype is configured on the Splunk side to parse the events properly. If you do not specify the sourcetype here, ensure that the default sourcetype associated with the Splunk HEC Token being used here is also configured on the Splunk side for proper parsing of events. If you cannot make this configuration, ask your Splunk Admin to make the configuration for the sourcetype. Refer section: [Configure sourcetypes for events](#)
- **Host:** The hostname or IP address of the network device that generated an event.
- **Database (*Either Database and Table Name or Advanced Query are Mandatory Fields.*):** The Databricks Database whose table needs to be used.
- **Table Name (*Either Database and Table Name or Advanced Query are Mandatory Fields.*):** Table from which the data to be pushed is obtained. The data is obtained by the query : `select \* from <database>.<table>`. If you want a different form of query, use the `Advanced query` parameter.
- **Filter:** Any filter or queries to run on top of the table specified.
- **Advanced Query (*Either Database and Table Name or Advanced Query are Mandatory Fields.*):** Complete query whose results you want to push to Splunk. If Advanced Query, Database, Table and Filter all are specified, the Advanced Query takes precedence and the other 3 parameters are ignored. The events to be pushed to Splunk are obtained from the Advanced Query in that case. However, if the Advanced query parameter is empty, the rows from the specified table based on the specified Filter are pushed to Splunk.

To run the notebook, **attach it to a cluster**, fill in all the required parameters and select the **Run All** option. In case of any error, the error is displayed at the bottom of the notebook cells where it occurred.

## Import Splunk Instance's Certificate in Databricks

This step is needed only if your Splunk instance has been configured to use a certificate. Follow the below steps to import the certificate.

- Create an init script as follows in Databricks. Here, PEM format CA certificates are added to the file myca.crt which is located at /usr/local/share/ca-certificates/.

**<YOUR CERTIFICATE CONTENT>** can be obtained from Splunk Admin.

```
#!/bin/bash

cat << 'EOF' > /usr/local/share/ca-certificates/myca..crt
-----BEGIN CERTIFICATE-----
<YOUR CERTIFICATE CONTENT>
-----END CERTIFICATE-----
EOF
```

```

update-ca-certificates

PEM_FILE="/etc/ssl/certs/myca.pem"
PASSWORD="<password>"
KEYSTORE="/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/security/cacerts"

CERTS=$(grep 'END CERTIFICATE' $PEM_FILE | wc -l)

# To process multiple certs with keytool, you need to extract
# each one from the PEM file and import it into the Java KeyStore.

for N in $(seq 0 $((CERTS - 1))); do
  ALIAS="$(basename $PEM_FILE)-$N"
  echo "Adding to keystore with alias:$ALIAS"
  cat $PEM_FILE |
    awk "n==$N { print }; /END CERTIFICATE/ { n++ }" |
    keytool -noprompt -import -trustcacerts -alias $ALIAS -keystore
$KEYSTORE -storepass $PASSWORD
done

echo "export REQUESTS_CA_BUNDLE=/etc/ssl/certs/ca-certificates.crt" >>
/databricks/spark/conf/spark-env.sh

```

- Attach the init script to the cluster as a [cluster-scoped init script](#).
- Restart the cluster.

## Limitations

- This notebook doesn't provide a way of maintaining a checkpoint since it depends on the data that the user is collecting. Hence, this may cause data duplication in Splunk. If the intention is to run this notebook periodically we highly recommend implementing the checkpointing mechanism to avoid indexing the same data again in Splunk.

# References

- Self-signed certificates Splunk:  
<https://docs.splunk.com/Documentation/Splunk/8.1.1/Security/Howtoself-signcertificates>  
<https://docs.splunk.com/Documentation/Splunk/8.1.1/Security/HowtoprepareyoursignedcertificatesforSplunk>
- Splunk HEC:  
<https://docs.splunk.com/Documentation/Splunk/8.1.0/Data/UsetheHTTPEventCollector>
- Importing Splunk custom certificates to Databricks:  
<https://kb.databricks.com/python/import-custom-ca-cert.html>
- Databricks CLI setup:<https://docs.databricks.com/dev-tools/cli/index.html#>
- Databricks Secret scope:  
<https://docs.databricks.com/security/secrets/secret-scopes.html>  
<https://docs.databricks.com/security/secrets/secrets.html>