Hortonworks Data Platform

Using WebHDFS REST API

(Jan 14, 2013)

docs.hortonworks.com

Hortonworks Data Platform : Using WebHDFS REST API

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

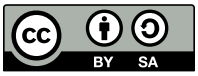

Table of Contents

1. Using WebHDFS REST API

This section provides information on using the WebHDFS REST APIs..

1.1. The WebHDFS REST API

Apache Hadoop provides native libraries for accessing HDFS. However, users prefer to use HDFS remotely over the heavy client side native libraries. For example, some applications need to load data in and out of the cluster, or to externally interact with the HDFS data. WebHDFS addresses these issues by providing a fully functional HTTP REST API to access HDFS.

WebHDFS provides the following features:

- Provides read and write access. Supports all HDFS operations (like granting permissions, configuring replication factor, accessing block location, etc.).
- Supports all HDFS parameters with defaults.
- Permits clients to access Hadoop from multiple languages without actually installing Hadoop. You can also use common tools like curl/wget to access HDFS.
- Uses the full bandwidth of the Hadoop cluster for streaming data: The file read and file write calls are redirected to the corresponding datanodes.
- Uses Kerberos (SPNEGO) and Hadoop delegation tokens for authentication.
- WebHDFS is completely Apache open source. Hortonworks contributed the code to Apache Hadoop as a first class built-in Hadoop component.
- Requires no additional servers. However, a proxy WebHDFS (for example: Httpfs is useful in certain cases and is complementary to WebHDFS).

For more information, see: WebHDFS – HTTP REST Access to HDFS.

In this section:

- User Guide
- Administrator Guide
- Resources

1.2. WebHDFS User Guide

The following examples use the `curl` command tool to access HDFS via WebHDFS REST API.

- To read a file (for example: `/foo/bar`):

```
curl -i -L "http://$<Host_Name>:$<Port>/webhdfs/v1/foo/bar?op=OPEN"
```

- To list a directory (for example: `/foo`):

```
curl -i "http://$<Host_Name>:$<Port>/webhdfs/v1/foo/?op=LISTSTATUS"
```

- To list the status of a file (for example: `/foo/bar`) or a directory:

```
curl -i "http://$<Host_Name>:$<Port>/webhdfs/v1/foo/bar?op=GETFILESTATUS"
```

- To write a file into a `/foo/new` file:

```
curl -i -X PUT -L "http://$<Host_Name>:$<Port>/webhdfs/v1/foo/newFile?op=
CREATE" -T newFile
```

- To rename the `/foo/bar` file to `/foo/bar3`:

```
curl -i -X PUT "http://$<Host_Name>:$<Port>/webhdfs/v1/foo/bar?op=RENAME&
destination=/foo/bar2"
```

- Make new directory `/foo2`:

```
curl -i -X PUT "http://$<Host_Name>:$<Port>/webhdfs/v1/foo2?op=MKDIRS&
permission=711"
```

1.2.1. Authentication

When security is enabled, authentication is performed by either Hadoop delegation token or Kerberos SPNEGO. If a token is set in the delegation query parameter, the authenticated user is the user encoded in the token. If the delegation parameter is not set, the user is authenticated by Kerberos SPNEGO.

Below are examples using the `curl` command tool.

- Login to the Key Distribution Center (KDC).

```
kinit
```

- Provide any arbitrary user name and a null password.

- Execute the following commands:

```
curl --negotiate -u:anyUser "http://$<Host_Name>:$<Port>/webhdfs/v1/foo/bar?
op=OPEN
```

```
curl --negotiate -u:anyUser -b ~/cookies.txt -c ~/cookies.txt http://
$<Host_Name>:$<Port>/webhdfs/v1/foo/bar?op=OPEN
```

where:

- `--negotiate` option enables SPNEGO in curl.
- `-u:anyUser` option is mandatory when the user name is not specified instead, a Kerberos established user (via kinit) is used. (Ensure that you provide any user name and enter a null password when prompted.)
- `-b` and `-c` options are used for storing and sending HTTP cookies.

1.3. WebHDFS Administrator Guide

Step 1: Set up WebHDFS.

Add the following property to the `hdfs-site.xml` file:

```
<property>
   <name>dfs.webhdfs.enabled</name>
   <value>true</value>
</property>
```

Step 2 (Conditional): If running a secure cluster, follow the steps listed below.

Step 2-a: Create an HTTP service user principal using the command given below:

```
kadmin: addprinc -randkey HTTP/$<Fully_Qualified_Domain_Name>@$<Realm_Name>.
COM
```

where:

- `Fully_Qualified_Domain_Name`: Host where NameNode is deployed
- `Realm_Name`: Name of your Kerberos realm

Step 2-b: Create keytab files for the HTTP principals.

```
kadmin: xst -norandkey -k /etc/security/spnego.service.keytab HTTP/
$<Fully_Qualified_Domain_Name>
```

Step 2-c: Verify that the keytab file and the principal are associated with the correct service.

```
klist –k -t /etc/security/spnego.service.keytab
```

Step 2-d: Add the following properties to the `hdfs-site.xml` file.

```
<property>
   <name>dfs.web.authentication.kerberos.principal</name>
   <value>HTTP/$<Fully_Qualified_Domain_Name>@$<Realm_Name>.COM</value>
</property>
```

```
<property>
   <name>dfs.web.authentication.kerberos.keytab</name>
   <value>/etc/security/spnego.service.keytab</value>
</property>
```

where:

- `Fully_Qualified_Domain_Name`: Host where NameNode is deployed
- `Realm_Name`: Name of your Kerberos realm

Step 3: Restart the NameNode and DataNode services.

- For HMC, see Step 1 in Starting HDP Services.

1.4. WebHDFS Resources

The HTTP REST API supports the complete FileSystem interface for HDFS. For more information, see the following sections in the WebHDFS REST API documentation:

- Introduction
  - Operations
  - FileSystem URIs vs. HTTP URLs
  - HDFS Configuration Options
- Authentication
- Proxy Users
- File and Directory Operations
- Other File System Operations
- Delegation Token Operations
- Error Responses
- JSON Schemas
- HTTP Query Parameter Dictionary