Hortonworks Data Platform

Apache Zeppelin Component Guide

(December 15, 2017)

docs.cloudera.com

Hortonworks Data Platform: Apache Zeppelin Component Guide

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.

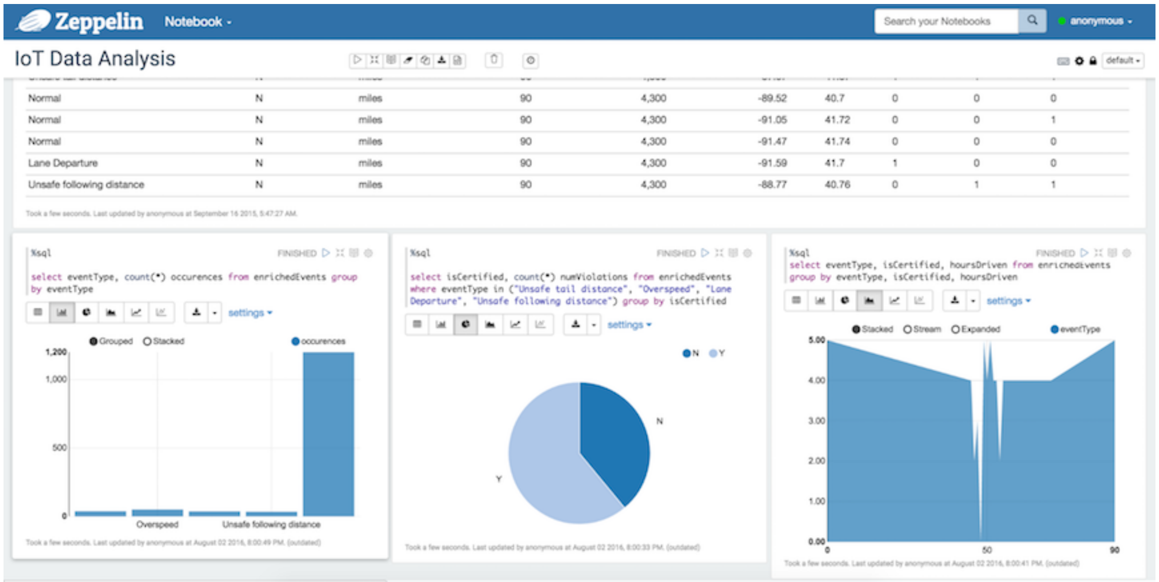
Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

1. Overview

Apache Zeppelin is a web-based notebook that supports interactive data exploration, visualization, and collaboration. Zeppelin supports a growing list of programming languages and interfaces, including Python, Scala, Hive, SparkSQL, shell, AngularJS, and markdown.

Apache Zeppelin is useful for working interactively with long workflows: developing, organizing, and running analytic code and visualizing results.



IoT Data Analysis

Normal	N	miles	90	4,300	-89.52	40.7	0	0	0
Normal	N	miles	90	4,300	-91.05	41.72	0	0	1
Normal	N	miles	90	4,300	-91.47	41.74	0	0	0
Lane Departure	N	miles	90	4,300	-91.59	41.7	1	0	0
Unsafe following distance	N	miles	90	4,300	-88.77	40.76	0	1	1

Took a few seconds. Last updated by anonymous at September 16 2015, 5:47:27 AM.

```
%sql

select eventType, count(*) occurences from enrichedEvents group
by eventType
```

FINISHED

Took a few seconds. Last updated by anonymous at August 02 2016, 8:00:49 PM. (outdated)

```
%sql

select isCertified, count(*) numViolations from enrichedEvents
where eventType in ("Unsafe tail distance", "Overspeed", "Lane
Departure", "Unsafe following distance") group by isCertified
```

FINISHED

Took a few seconds. Last updated by anonymous at August 02 2016, 8:00:33 PM. (outdated)

```
%sql
select eventType, isCertified, hoursDriven from enrichedEvents
group by eventType, isCertified, hoursDriven
```

FINISHED

Took a few seconds. Last updated by anonymous at August 02 2016, 8:00:41 PM. (outdated)

2. Installing Apache Zeppelin

This chapter describes how to install Zeppelin on an Ambari-managed cluster.

To install Zeppelin on an HDP cluster not managed by Ambari, see Installing and Configuring Apache Zeppelin in the *Command Line Installation Guide*.

To configure Zeppelin security features, see Configuring Zeppelin Security.

2.1. Requirements for Installing Zeppelin

Install Zeppelin on a node where Spark clients are already installed and running. This typically means that Zeppelin will be installed on a gateway or edge node.

Zeppelin requires the following software versions:

- HDP 2.6.0 or later.
- Apache Spark version 1.6 or Spark 2.0.
- Java 8 on the node where Zeppelin is installed.

The optional Livy server provides security features and user impersonation support for Zeppelin users. Livy is installed as part of Spark.

- For an Ambari-managed cluster, see Installing Spark Using Ambari.

  After installing Spark, Livy, and Zeppelin, refer to Configuring Zeppelin for post-installation steps.
- For a cluster not managed by Ambari, see Installing and Configuring Livy in the *Command Line Installation Guide*.

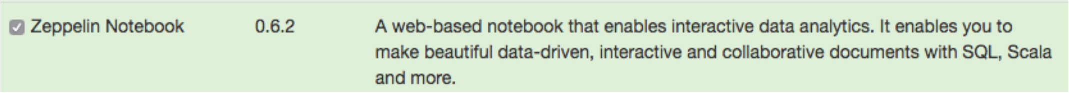
2.2. Installing Zeppelin Using Ambari

This section describes how to install Zeppelin using Ambari.

The Ambari installation wizard sets default values for Zeppelin configuration settings. At first you should accept the default settings. Later, when you are more familiar with Zeppelin, consider customizing Zeppelin configuration settings.

To install Zeppelin using Ambari, add the Zeppelin service:

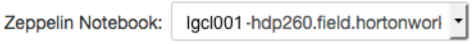
1. In the "Actions" menu at the bottom left of the Ambari dashboard, select "Add Service". This starts the Add Service wizard, which displays the "Choose Services" page.
2. On the Choose Services page, select the Zeppelin service:



☑ Zeppelin Notebook 0.6.2 A web-based notebook that enables interactive data analytics. It enables you to make beautiful data-driven, interactive and collaborative documents with SQL, Scala and more.

3. Click "Next" to continue.

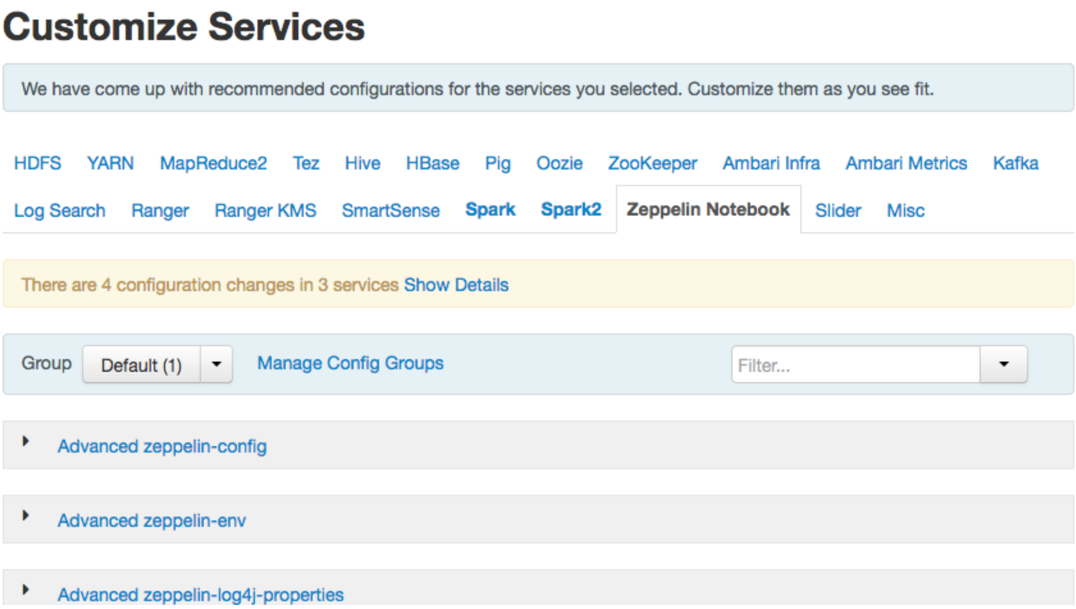
4. On the Assign Masters page, review the node assignment for Zeppelin, and modify as needed:



Zeppelin Notebook: lgcl001-hdp260.field.hortonworl

5. Click "Next" to continue.

6. On the Customize Services page, review default values and click "Next" to continue.



Customize Services

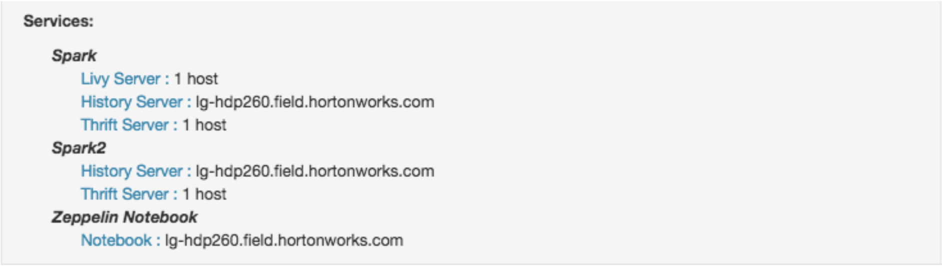
We have come up with recommended configurations for the services you selected. Customize them as you see fit.

There are 4 configuration changes in 3 services Show Details

Group: Default (1)

7. If Kerberos is enabled on the cluster, review principal and keytab settings on the Configure Identities page, modify settings if desired, and then click Next.

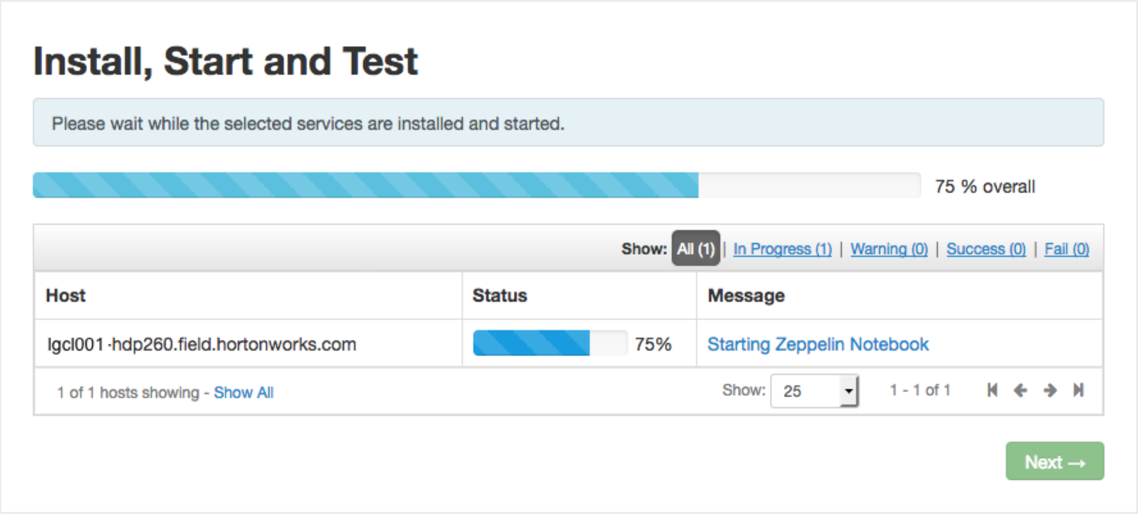
8. On the Review page, scroll down to the "Services" section and confirm the node selected to run the Zeppelin service; for example:



Services:

Spark
Livy Server : 1 host
History Server : lg-hdp260.field.hortonworks.com
Thrift Server : 1 host
Spark2
History Server : lg-hdp260.field.hortonworks.com
Thrift Server : 1 host
Zeppelin Notebook
Notebook : lg-hdp260.field.hortonworks.com

9. Click "Deploy" to complete the installation process. The Install, Start and Test page shows status as deployment proceeds:



Install, Start and Test

Please wait while the selected services are installed and started.

75 % overall

Show: All (1) | In Progress (1) | Warning (0) | Success (0) | Fail (0)

Host	Status	Message
lgcl001-hdp260.field.hortonworks.com	75%	Starting Zeppelin Notebook

1 of 1 hosts showing - Show All

Show: 25
1 - 1 of 1
Next →

To validate your Zeppelin installation, open the Zeppelin Web UI in a browser window. Use the port number configured for Zeppelin (9995 by default); for example:

```
http://<zeppelin-host>:9995
```

To check the Zeppelin version number, type the following command on the command line:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh --version
```

Zeppelin stores configuration settings in the `/etc/zeppelin/conf` directory. Note, however, that if your cluster is managed by Ambari you should not modify configuration settings directly. Instead, use the Ambari web UI.

Zeppelin stores log files in `/var/log/zeppelin` on the node where Zeppelin is installed.

3. Configuring Zeppelin

This chapter describes how to configure the following optional features:

- Configuring Livy on an Ambari-managed cluster. (For a cluster not managed by Ambari, see Installing and Configuring Livy in the *Command Line Installation Guide*.)
- Configuring Zeppelin to access Hive through the `%jdbc` interpreter

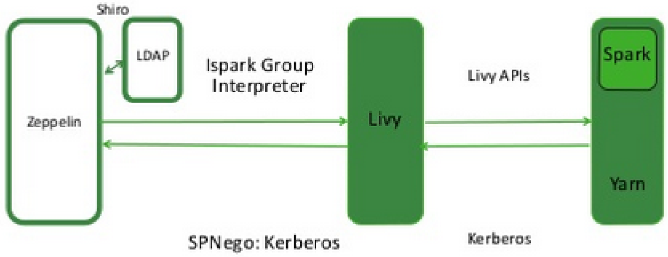
3.1. Configuring Livy on an Ambari-Managed Cluster

Livy is a proxy service for Apache Spark; it offers the following capabilities:

- Zeppelin users can launch a Spark session on a cluster, submit code, and retrieve job results, all over a secure connection.
- When Zeppelin runs with authentication enabled, Livy propagates user information when a session is created. Livy user impersonation offers an extended multi-tenant experience, allowing users to share RDDs and cluster resources. Multiple users can access their own private data and session, and collaborate on a notebook.

Note: Livy supports Kerberos, but does not require it.

The following graphic shows process communication among Zeppelin, Livy, and Spark:



On an Ambari-managed cluster, Livy is installed with Spark.

Here are several optional configuration steps:

- (Kerberos-enabled clusters) Ensure that access is enabled only for groups and hosts where Livy runs.
- Check the Livy host URL:
  1. Navigate to the Interpreter configuration page in the Zeppelin Web UI.
  2. In the livy interpreter section, make sure that the `zeppelin.livy.url` property contains the full Livy host name; replace `localhost` if necessary.
  3. Scroll down to the "Save" button, and click "Save".

  Note: on an Ambari-managed cluster you can find the Livy host from the Ambari dashboard: navigate to Spark > Summary > Livy Server.

- Configure Livy impersonation:

  1. From the Ambari dashboard, navigate to Spark > Configs.

  2. Open the "Custom livy-conf" category.

  3. Ensure that `livy.superusers` is listed; if not, add the property.

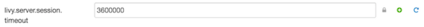
  4. Set `livy.superusers` to the user account associated with Zeppelin, `zeppelin.livy.principal`.

     For example, if `zeppelin.livy.principal` is `zeppelin-sr1@example.com`, set `livy.superusers` to the same account, `zeppelin-sr1@example.com`.

- Specify a timeout value for Livy sessions.

  By default, Livy preserves cluster resources by recycling sessions after one hour of session inactivity. When a Livy session times out, the Livy interpreter must be restarted.

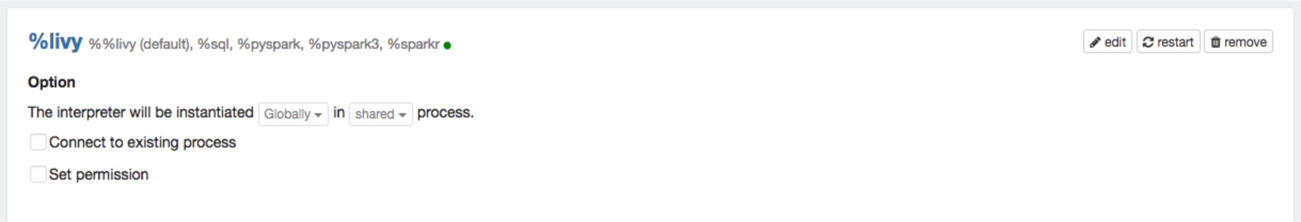
  To specify a larger or smaller value using Ambari, navigate to the `livy.server.session.timeout` property in the "Advanced livy-conf" section of the Spark service. Specify the timeout in milliseconds (the default is 3600000, one hour):



livy.server.session.timeout 3600000

If you change any Livy interpreter settings, restart the Livy interpreter.

Navigate to the Interpreter configuration page in the Zeppelin Web UI. Locate the Livy interpreter and click "restart":



%livy %%livy (default), %sql, %pyspark, %pyspark3, %sparkr ●

Option

The interpreter will be instantiated Globally in shared process.

□ Connect to existing process

□ Set permission

To verify that the Livy server is running, access the Livy web UI in a browser window. The default port is 8998:

```
http://<livy-hostname>:8998/
```

3.2. Configuring User Impersonation for Access to Hive

User impersonation runs Hive queries under the user ID associated with the Zeppelin session.

If Kerberos is not enabled on the cluster, complete the following steps:

1. In the Zeppelin UI, navigate to the `%jdbc` section of the Interpreter page.
2. Click **edit**, then add a `hive.proxy.user.property` property and set its value to `hive.server2.proxy.user`.
3. Click **Save**, then click **restart** to restart the JDBC interpreter.

If Kerberos is enabled on the cluster, enable user impersonation as follows:

To configure the `%jdbc` interpreter, complete the following steps:

1. In Hive configuration settings, set `hive.server2.enable.doAs` to `true`.
2. In the Zeppelin UI, navigate to the `%jdbc` section of the Interpreter page.
3. Enable authentication via the Shiro configuration: specify authorization type, keytab, and principal.

   a. Set `zeppelin.jdbc.auth.type` to KERBEROS.

   b. Set `zeppelin.jdbc.principal` to the value of the principal.

   c. Set `zeppelin.jdbc.keytab.location` to the keytab location.
4. Set `hive.url` to the URL for HiveServer2. (On an Ambari-managed cluster you can find the URL under Hive > HiveServer2 JDBC URL.) Here is the general format:

   ```
   jdbc:hive2://HiveHost:10001/default;principal=hive/_HOST@HOST1.COM;hive.
   server2.proxy.user=testuser
   ```

   The JDBC interpreter adds the user ID as a proxy user, and sends the string to HiveServer2; for example:

   ```
   jdbc:hive2://dkhdp253.dk:2181,dkhdp252.dk:2181,dkhdp251.dk:2181/
   ;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2
   ```
5. Add a `hive.proxy.user.property` property and set its value to `hive.server2.proxy.user`.
6. Click **Save**, then click **restart** to restart the interpreter.

For information about authenticating Zeppelin users through Active Directory or LDAP, see Configuring Zeppelin Security.

3.3. Configuring User Impersonation for Access to Phoenix

User impersonation runs Phoenix queries under the user ID associated with the Zeppelin session.

To enable user impersonation for Phoenix, complete the following steps in the Ambari dashboard:

1. In HBase configuration settings, enable phoenix sql.

2. In Advanced HBase settings, set the following properties:

   ```
   hbase.thrift.support.proxyuser=true
   hbase.regionserver.thrift.http=true
   ```

3. In HDFS configuration settings, set the following properties:

   ```
   hadoop.proxyuser.hbase.groups=*
   hadoop.proxyuser.hbase.hosts=*
   hadoop.proxyuser.zeppelin.groups=*
   hadoop.proxyuser.zeppelin.hosts=*
   ```

4. Make sure that the user has access to HBase. You can verify this from the HBase shell with the `user_permissions` command.

4. Using Zeppelin

A Zeppelin notebook is a browser-based GUI for interactive data exploration, modeling, and visualization.

As a notebook author or collaborator, you write code in a browser window. When you run the code from the browser, Zeppelin sends the code to backend processors such as Spark. The processor or service and returns results; you can then use Zeppelin to review and visualize results in the browser.

Note: Zeppelin on HDP does not support sharing a note by sharing its URL, due to lack of proper access control over how and with whom a note can be shared.

Apache Zeppelin is supported on the following browsers:

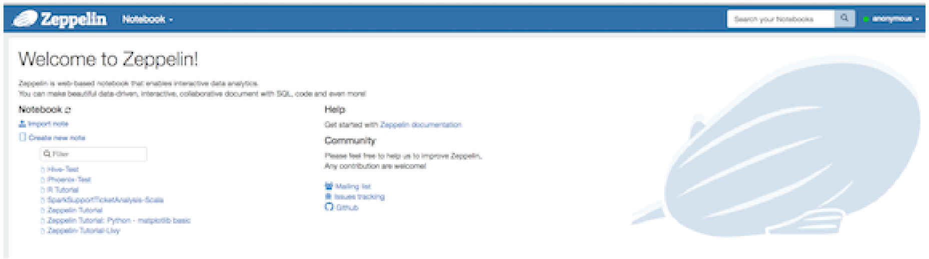
- Internet Explorer, latest supported releases. (Zeppelin is not supported on versions 8 or 9, due to lack of native support for WebSockets.)
- Google Chrome, latest stable release.
- Mozilla Firefox, latest stable release.
- Apple Safari, latest stable release. Note that when SSL is enabled for Zeppelin, Safari requires a Certificate Authority-signed certificate to access the Zeppelin UI.

4.1. Launching Zeppelin

To launch Zeppelin in your browser, access the host and port associated with the Zeppelin server. The default port is 9995:

`http://<zeppelinhost>:9995`.

When you first connect to Zeppelin, you will see the home page:



Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Get started with Zeppelin documentation

Please feel free to help us to improve Zeppelin.
Any contribution are welcome!

If Zeppelin is configured for LDAP or Active Directory authentication, log in before using Zeppelin:

1. Click the Login button at the top right corner of the page.
2. In the login dialog box, specify a valid username and password.

   If Active Directory is used for the identity store, you might need to fully qualify your account name (unless the `activeDirectoryRealm.principalSuffix` property was set during AD configuration); for example:

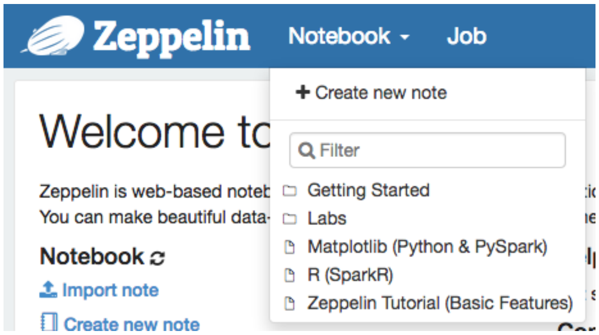
```
ad-username@AD.DOMAIN.COM
```

3. Zeppelin presents its home page.

The following menus are available in the top banner of all Zeppelin pages:

- **Notebook**

  Open a note, filter the list of notes by name, or create a note:

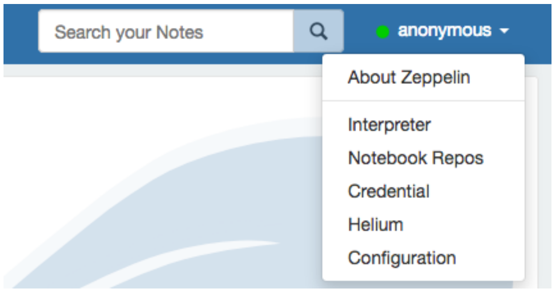


Zeppelin is web-based noteb

You can make beautiful data-

- **User settings**

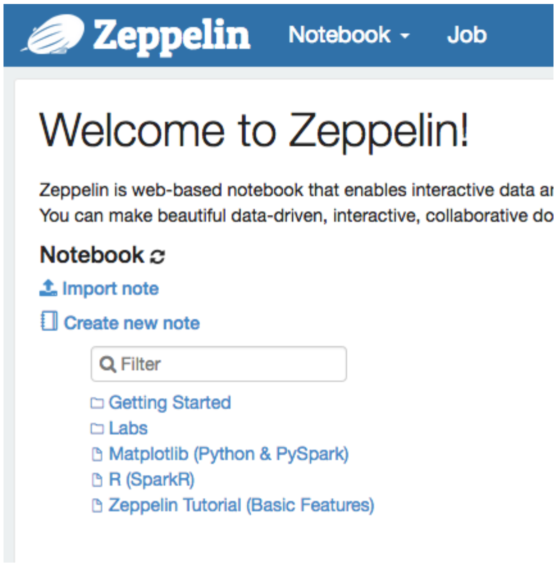
  Displays your username, or "anonymous" if security is not configured for Zeppelin.



  - List version information about Zeppelin
  - Review interpreter settings and configure, add, or remove interpreter instances
  - Save credentials for data sources
  - Display configuration settings

Each instance of Zeppelin contains "notes", which are the fundamental elements of a Zeppelin notebook.

Zeppelin lists available notes on the left side of the welcome screen and in the "Notebook" menu. Zeppelin ships with several sample notes, including tutorials:



Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data ar
You can make beautiful data-driven, interactive, collaborative do

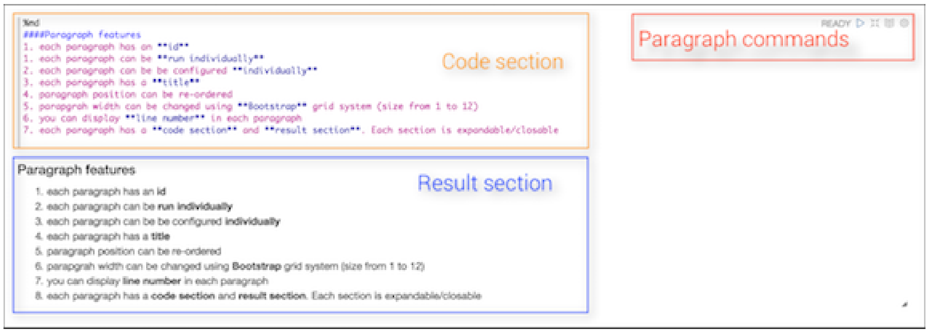
Notebook

Import note
Create new note
Filter
Getting Started
Labs
Matplotlib (Python & PySpark)
R (SparkR)
Zeppelin Tutorial (Basic Features)

4.2. Working with Notes

A note consists of one or more paragraphs of code, which you can use to define and run snippets of code in a flexible manner.

A paragraph contains code to access services, run jobs, and display results. A paragraph consists of two main sections: an interactive box for code, and a box that displays results. To the right is a set of paragraph commands. The following graphic shows paragraph layout.

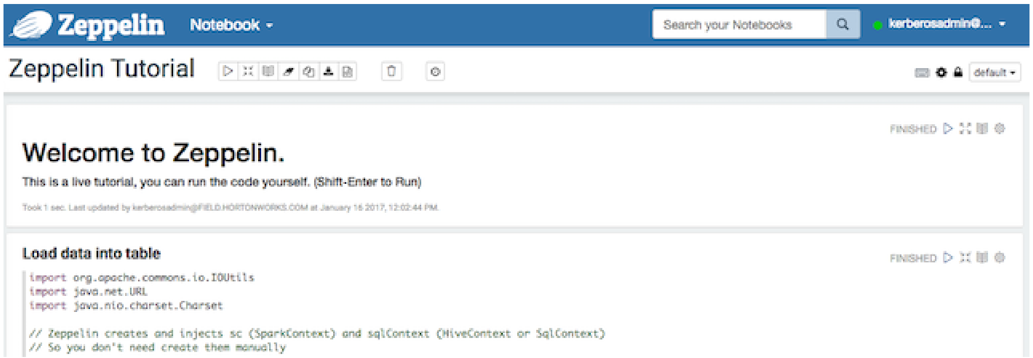


```
%md
####Paragraph features
1. each paragraph has an **id**
1. each paragraph can be **run individually**
2. each paragraph can be be configured **individually**
3. each paragraph has a **title**
4. paragraph position can be re-ordered
5. parapgrah width can be changed using **Bootstrap** grid system (size from 1 to 12)
6. you can display **line number** in each paragraph
7. each paragraph has a **code section** and **result section**. Each section is expandable/closable
```

Paragraph features

1. each paragraph has an **id**
2. each paragraph can be **run individually**
3. each paragraph can be be configured **individually**
4. each paragraph has a **title**
5. paragraph position can be re-ordered
6. parapgrah width can be changed using **Bootstrap** grid system (size from 1 to 12)
7. you can display **line number** in each paragraph
8. each paragraph has a **code section** and **result section**. Each section is expandable/closable

Zeppelin ships with several sample notes, including tutorials that demonstrate how to run Spark scala code, Spark SQL code, and create visualizations.



Zeppelin Tutorial

Welcome to Zeppelin.

This is a live tutorial, you can run the code yourself. (Shift-Enter to Run)

Took 1 sec. Last updated by kerberosadmin@FIELD.HORTONWORKS.COM at January 16 2017, 12:02:44 PM.

Load data into table

```
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset

// Zeppelin creates and injects sc (SparkContext) and sqlContext (HiveContext or SqlContext)
// So you don't need create them manually
```

To run a tutorial:

1. Navigate to the tutorial: click one of the Zeppelin tutorial links on the left side of the welcome page, or use the Notebook pull-down menu.

2. Zeppelin presents the tutorial, a sequence of paragraphs prepopulated with code and text.

3. Starting with the first paragraph, click the triangle button at the upper right of the paragraph. The status changes to PENDING, RUNNING, and then FINISHED when done.

4. When the first cell finishes execution, results appear in the box underneath your code. Review the results.

5. Step through each cell, running the code and reviewing results.

4.2.1. Creating and Running a Note

To create a note:

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.

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.

To run your code:

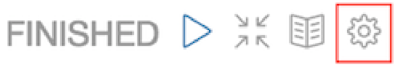
1. Click the triangle button in the cell that contains your code:

   ▷

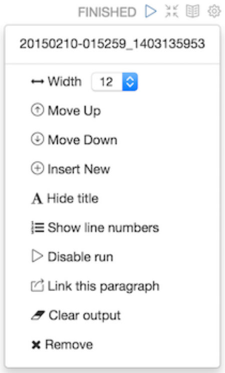
2. Zeppelin displays status near the triangle button: PENDING, RUNNING, ERROR, or FINISHED.

3. When finished, results appear in the result section below your code.

The settings icon (outlined in red) offers several additional commands:



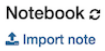
These commands allow you to perform several note operations, such as showing and hiding line numbers, clearing the results section, and deleting the paragraph.



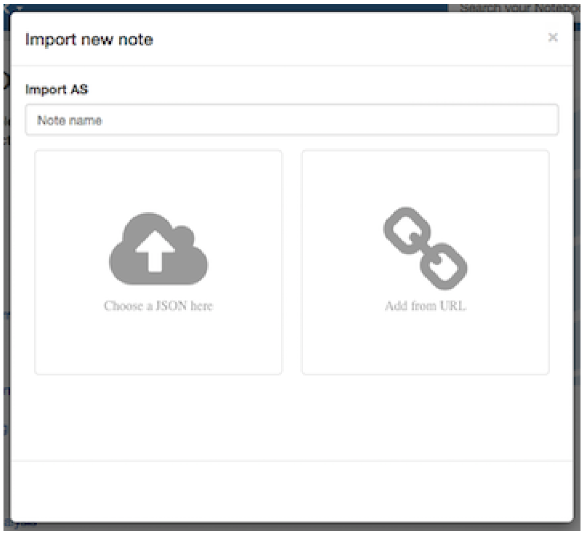
4.2.2. Importing a Note

To import a note from a URL or from a JSON file in your local file system:

1. Click "Import note" on the Zeppelin home page:



2. Zeppelin displays an import dialog box:

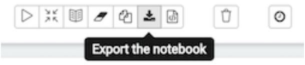


3. To upload the file or specify the URL, click the associated box.

By default, the name of the imported note is the same as the original note. You can rename it by providing a new name in the "Import AS" field.

4.2.3. Exporting a Note

To export a note to a local JSON file, use the export note icon in the note toolbar:



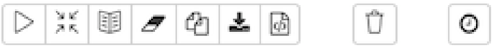
Zeppelin downloads the note to the local file system.

Note: Zeppelin exports code and results sections in all paragraphs. If you have a lot of data in your results sections, consider trimming results before exporting them.

4.2.4. Using the Note Toolbar

At the top of each note there is a toolbar with buttons for running code in paragraphs and for setting configuration, security, and display options:

There are several buttons in the middle of the toolbar:

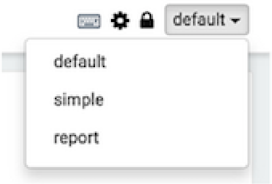


These buttons perform the following operations:

- Execute all paragraphs in the note sequentially, in the order in which they are displayed in the note.
- Hide or show the code section of all paragraphs.
- Hide or show the result sections in all paragraphs.
- Clear the result section in all paragraphs.
- Clone the current note.
- Export the current note to a JSON file.

  Note that the code and result sections in all paragraphs are exported. If you have extra data in some of your result sections, trim the data before exporting it.
- Commit the current note content.
- Delete the note.
- Schedule the execution of all paragraphs using CRON syntax. This feature is not currently operational. If you need to schedule Spark jobs, consider using Oozie Spark action.

There are additional buttons on the right side of the toolbar:



These buttons perform the following operations (from left to right):

- Display all keyboard shortcuts.
- Configure interpreters that are bound to the current note.
- Configure note permissions.
- Switch display mode:
    - *Default*: the notebook can be shared with (and edited by) anyone who has access to the notebook.
    - *Simple*: similar to default, with available options shown only when your cursor is over the cell.
    - *Report*: only your results are visible, and are read-only (no editing).

    Note: Zeppelin on HDP does not support sharing a note by sharing its URL, due to lack of proper access control over who and how a note can be shared.

4.3. Importing External Packages

If you want to use an external package within a Zeppelin note, you can follow one of these approaches:

- Specify the dependency for the associated interpreter on the Interpreter page.

    For more information, see Dependency Management for Interpreter at `zeppelin.apache.org`.

- For Spark jobs, you can pass a jar, package, or list of files to `spark-submit` using SPARK_SUBMIT_OPTIONS; for example:
    - SPARK_SUBMIT_OPTIONS in `conf/zeppelin-env.sh`

        ```
        export SPARKSUBMITOPTIONS="--packages com.databricks:spark-csv_2.10:1.2.0
         --jars /path/mylib1.jar,/path/mylib2.jar --files /path/mylib1.py,/path/
        mylib2.zip,/path/mylib3.egg"
        ```

    - In SPARK_HOME/`conf/spark-defaults.conf`

        ```
        spark.jars /path/mylib1.jar,/path/mylib2.jar spark.jars.packages com.
        databricks:spark-csv_2.10:1.2.0 spark.files /path/mylib1.py,/path/mylib2.
        egg,/path/mylib3.zip
        ```

If you want to import a library for a note that uses the Livy interpreter, see Using the `%livy` Interpreter to Access Spark.

4.4. Configuring and Using Zeppelin Interpreters

An interpreter is a plugin that enables you to access processing engines and data sources from the Zeppelin UI.

For example, if you want to use Python code in your Zeppelin notebook, you need a Python interpreter. Each interpreter runs in its own JVM on the same node as the Zeppelin server. The Zeppelin server communicates with interpreters through the use of Thrift.

Apache Zeppelin on HDP supports the following interpreters:

- Spark
- JDBC (supports Hive, Phoenix)
- OS Shell
- Markdown
- Livy (supports Spark, Spark SQL, PySpark, PySpark3, and SparkR)
- AngularJS

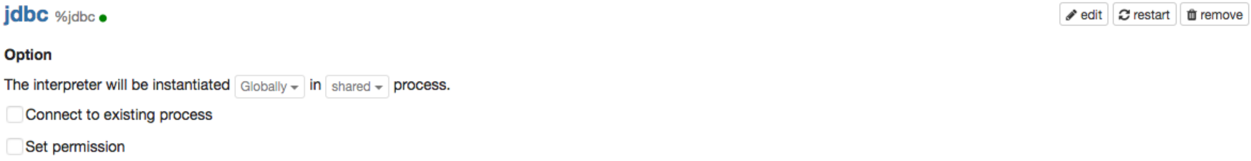
Note: PySpark and associated libraries require Python version 2.7 or later, or Python version 3.4 or later, installed on all nodes.

4.4.1. Modifying Interpreter Settings

Before using an interpreter, you might want to modify default settings such as the home directory for the Spark interpreter, the name of the Hive JDBC driver for the JDBC interpreter, or the keytab and principal name for a secure installation.

To set custom configuration values for an interpreter:

1. Click the user settings menu and navigate to the Interpreter page.
2. Scroll down to the Properties section for the interpreter of interest, and click "edit":



jdbc %jdbc ●

Option

The interpreter will be instantiated Globally in shared process.

□ Connect to existing process

□ Set permission

3. Make your changes.
4. Scroll to the end of the list of settings and click "Save".
5. Some types of changes require restarting the interpreter; use the button next to the edit button.

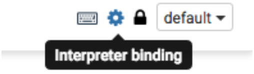
Note: The Interpreter page is subject to access control settings. If the page does not list a set of interpreters, check with your system administrator.

4.4.2. Using Zeppelin Interpreters

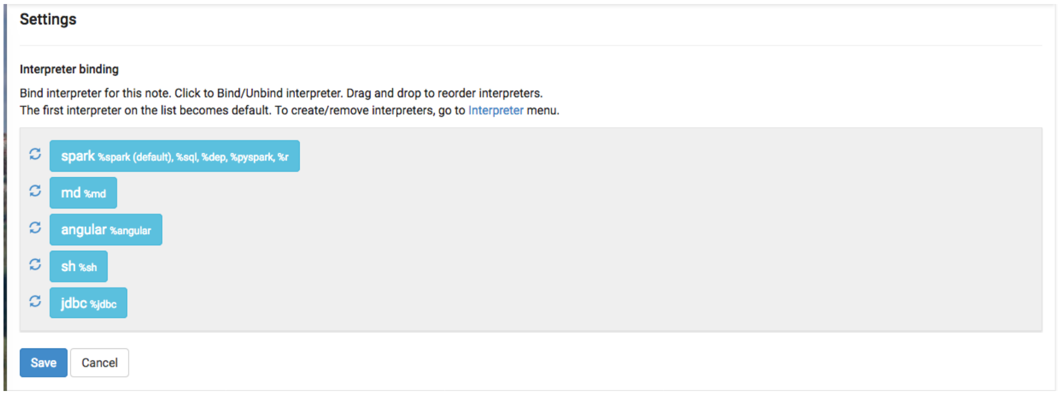
Before using an interpreter, ensure that the interpreter is available for use in your note:

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 reorder interpreters.
The first interpreter on the list becomes default. To create/remove interpreters, go to Interpreter menu.

spark %spark (default), %sql, %dep, %pyspark, %r

md %md

angular %angular

sh %sh

jdbc %jdbc

4. To select an interpreter, click on the interpreter name to select the interpreter. Each click operates as a toggle.

5. 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".

Using 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.

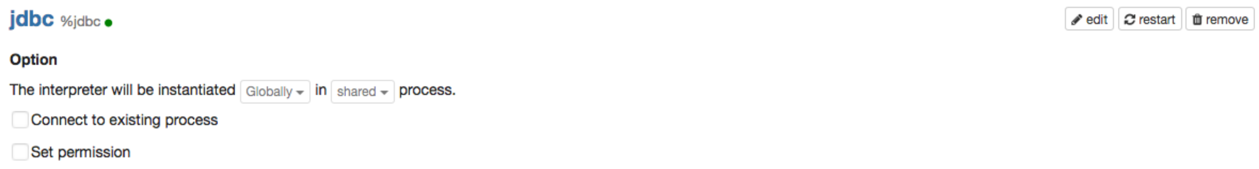
The following paragraph uses the `%sh` interpreter to access the system shell and list the current working directory:

```
%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.

Using 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.

For more information about interpreters, see Interpreters in Apache Zeppelin at zeppelin.apache.org.

4.4.3. Using the %jdbc Interpreter to Access Hive

The `%jdbc` interpreter supports access to Apache Hive data. The interpreter connects to Hive via Thrift.

If you want Hive queries to run under the user ID originating the query, see Configuring User Impersonation for Access to Hive.

To use the JDBC interpreter to access Hive:

1. Add the following directive at the start of a paragraph:

   ```
   %jdbc(hive)
   ```

2. Next, add the query that accesses Hive.

Here is a sample paragraph:

```
%jdbc(hive)
SELECT * FROM db_name;
```

If you receive an error, you might need to complete the following additional steps:

1. Copy Hive jar files to `/usr/hdp/current/zeppelin-server/interpreter/jdbc` (or create a soft link).

2. In the Zeppelin UI, navigate to the `%jdbc` section of the Interpreter page.

3. Click **edit**, then add a `hive.proxy.user.property` property and set its value to `hive.server2.proxy.user`.

4. Click **Save**, then click **restart** to restart the JDBC interpreter.

4.4.4. Using the `%jdbc` Interpreter to Access Phoenix

The `%jdbc` interpreter supports access to Apache Phoenix data.

If you want Phoenix queries to run under the user ID originating the query, see Configuring User Impersonation for Access to Phoenix.

To use the JDBC interpreter to access Phoenix:

1. Add the following directive at the start of a paragraph:

   ```
   %jdbc(phoenix)
   ```

2. Run the query that accesses Phoenix.

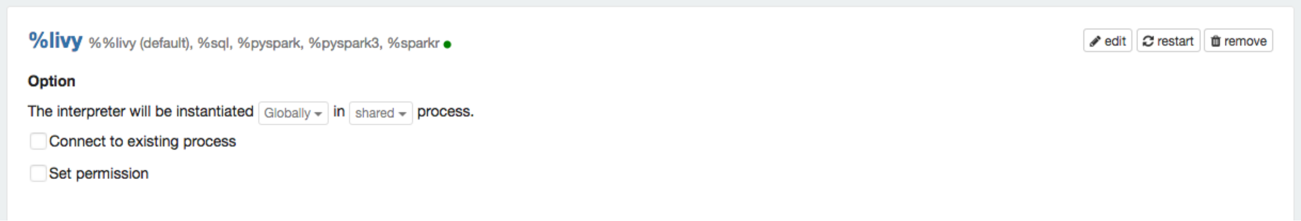
4.4.5. Using the `%livy` Interpreter to Access Spark

The Livy interpreter offers several advantages over the default Spark interpreter (`%spark`):

- Sharing of Spark context across multiple Zeppelin instances.
- Reduced resource use, by recycling resources after 60 minutes of activity (by default). The default Spark interpreter runs jobs–and retains job resources–indefinitely.
- User impersonation. When the Zeppelin server runs with authentication enabled, the Livy interpreter propagates user identity to the Spark job so that the job runs as the originating user. This is especially useful when multiple users are expected to connect to the same set of data repositories within an enterprise. (The default Spark interpreter runs jobs as the default Zeppelin user.)
- The ability to run Spark in yarn-cluster mode.

Prerequisites:

- Before using SparkR through Livy, R must be installed on all nodes of your cluster. For more information, see SparkR Prerequisites in the *Spark Component Guide*.
- Before using Livy in a note, check the Interpreter page to ensure that the Livy interpreter is configured properly for your cluster:



%livy %%livy (default), %sql, %pyspark, %pyspark3, %sparkr ●

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 access settings, check with your system administrator for more information.

To access PySpark using Livy, specify the corresponding interpreter directive before the code that accesses Spark; for example:

```
%livy.pyspark
print "1"

1
```

Similarly, to access SparkR using Livy, specify the corresponding interpreter directive:

```
%livy.sparkr
hello <- function( name ) {
    sprintf( "Hello, %s", name );
}

hello("livy")
```

Important

To use SQLContext with Livy, do not create SQLContext explicitly. Zeppelin creates SQLContext by default.

If necessary, remove the following lines from the SparkSQL declaration area of your note:

```
//val sqlContext = new org.apache.spark.sql.SQLContext(sc)
//import sqlContext.implicits._
```

Livy sessions are recycled after a specified period of session inactivity. The default is one hour.

For more information about using Livy with Spark 1 or Spark 2, see Submitting Spark Applications Through Livy.

Importing External Packages

To import an external package for use in a note that runs with Livy:

1. Navigate to the interpreter settings.

2. If you are running the Livy interpreter in local mode (as specified by `livy.spark.master`), add jar files to the `/usr/hdp/<version>/livy/repl-jars` directory.

3. If you are running the Livy interepreter in yarn-cluster mode, either complete step 2 or edit the Livy configuration on the Interpreters page as follows:

   a. Add a new key, `livy.spark.jars.packages`.

   b. Set its value to `<group>:<id>:<version>`.

      Here is an example for the `spray-json` library, which implements JSON in Scala:

      `io.spray:spray-json_2.10:1.3.1`

Running Applications that Require Third-party Libraries

If you want to run Spark applications that require third-party libraries through Zeppelin/ Livy, complete the following additional step so that the libraries are available on all nodes:

In the Zeppelin UI, click the user settings menu at the top right, then select Interpreter. On the Interpreters page, set the following livy.spark.jars livy property to point to the applicable .jar files. This must be an HDFS location.

5. Configuring Zeppelin Security

Zeppelin uses Apache Shiro to provide authentication and authorization (access control).

This chapter describes how to configure and enable several Zeppelin security features:

1. Configure authentication. Zeppelin supports a Shiro-based identity source for testing and informal use, as well as LDAP and Active Directory identity sources for production use. After authentication is enabled, when users connect to Apache Zeppelin they are prompted for login credentials.
2. Optionally, limit who can configure Zeppelin interpreter, credential, and configuration settings; notebooks; and data.
3. Optionally, configure the Zeppelin UI to run over SSL (HTTPS).
4. Optionally, configure Zeppelin to run on a Kerberos-enabled cluster.

If Ranger is enabled on your cluster, no additional configuration steps are required to have Ranger work with Zeppelin. Note, however, that a Ranger policy change takes about five to ten minutes to take effect.

5.1. Getting Started

5.1.1. Prerequisites

To use LDAP or Active Directory (AD) as the identity store, LDAP or AD must be installed and running on your cluster. You will need LDAP or AD coordinates to configure them for use with Zeppelin. In addition, the associated user accounts must be defined on your Zeppelin nodes.

5.1.2. Configuring Security on an Ambari-Managed Cluster

If your cluster is managed by Ambari, navigate to the Configs tab and edit settings in the "Advanced zeppelin-env", "Advanced zeppelin-config", "zeppelin-log4j-properties" and "zeppelin-shiro-ini" sections, as described in following subsections.

Changes to `shiro_ini_content` require restarting the Zeppelin server. Ambari indicates this with a warning, and offers a menu option to restart Zeppelin.

5.1.3. Configuring Security on a Cluster Not Managed by Ambari

If your cluster is not managed by Ambari:

1. Locate the `shiro.ini` template file in the Zeppelin `/conf` folder:

   `/usr/hdp/current/zeppelin-server/conf/shiro.ini.template`.
2. Copy the template file as `shiro.ini`:

   `/usr/hdp/current/zeppelin-server/conf/shiro.ini`

3. Edit shiro.ini file contents, as described in the following subsections.

4. After editing the shiro.ini file, restart the Zeppelin server:

   ```
   ./bin/zeppelin-daemon.sh restart
   ```

5.1.4. shiro.ini Structure

The shiro_ini_content property (Ambari) and shiro.ini file (non-Ambari) contain several sections for configuring authentication:

- [main], which contains definitions for LDAP or Active Directory objects and properties.
- [users], which can be used to specify user accounts and passwords for simple deployments that do not require secure passwords, and require only a small number of statically-defined accounts.
- [roles], for defining roles associated with access control.
- [urls], for configuring URL-based security. For Zepplin, the [urls] section is used to specify authentication method and define access control filters.

For more information about Shiro configuration and processing, see Apache Shiro Configuration.

5.2. Configure Zeppelin for Authentication: Non-Production Use

The following steps provide a quick, basic form of authentication. This approach is not for production use; usernames and passwords are exposed in clear text. For production use, you should use LDAP or Active Directory as the identity source.

To configure authentication for informal use or testing:

1. Populate the [urls] section as follows:

   a. Specify authc as the authentication method in the URL section of shiro.ini contents, and make sure that the authc line is not commented out.

   b. To disable anonymous access to Zeppelin, add a comment character (#) at the start of the line containing /** = anon.

   Here is an example:

   ```
   [urls]
   #/api/version = anon
   #/** = anon
   /** = authc
   ```

2. Populate the [users] section as follows:

   Specify authorized accounts and associated passwords in shiro_ini settings: for clusters managed by Ambari, update shiro_ini_content; for non-Ambari clusters, update the shiro.ini file.

The following example configures authentication for users `admin`, `user1`, and `user2`, with passwords `password1`, `password2`, and `password3`, respectively:

```
[users]
admin = password1
user1 = password2
user2 = password3
```

3. Restart the Zeppelin server using Ambari or, for a cluster not managed by Ambari, follow the instructions in Installing and Configuring Apache Zeppelin in the *Non-Ambari Cluster Installation Guide*.

4. After completing these steps, Zeppelin requires authentication of user credentials before allowing access to the Zeppelin UI.

5.3. Configure Zeppelin for Authentication: LDAP and Active Directory

Zeppelin supports LDAP and Active Directory (AD) as identity stores for authentication. Because Active Directory is based on LDAP requirements, the configuration process is similar; however, the properties differ.

Before configuring LDAP or AD, user accounts must exist on Zeppelin nodes; and users, groups, and domain information must be stored in your LDAP or AD directory. You will need user, group, and domain information to configure LDAP or AD for Zeppelin.

Important

You should configure and enable SSL to the Zeppelin Web server whenever you enable LDAP or AD authentication. Without SSL, network communication is visible.

5.3.1. Configuring Authentication for Production Using Active Directory

To enable AD authentication, complete the following steps.

Note: Zeppelin currently uses Bind requests to authenticate end users; it does not support the LDAP `compare` operation.

The following steps describe basic settings. For more information about these and other settings, see Shiro Settings: Reference and Shiro authentication for Apache Zeppelin.

1. Secure the HTTP channel.

   In the `[urls]` section of `shiro.ini` contents, uncomment the line `/** = authc` and comment out the line `/** = anon` (to disable anonymous access):

```
[urls]
/api/version = anon
#/** = anon
/** = authc
```

Note: The [urls] section is processed from top to bottom; earlier statements have precedence. If you have two conflicting lines, the first is honored.

2. In the [main] section of shiro.ini contents, enable activeDirectoryRealm and modify the following settings for your operating environment. For clusters managed by Ambari, update shiro_ini_content; for non-Ambari clusters, update the shiro.ini file.

   Note that there are two types of directory references, those that refer to the AD database, and those that refer to user accounts and groups. Domain information can differ between the two.

   ```
   [main]

   # authentication settings
   activeDirectoryRealm = org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
   activeDirectoryRealm.url = ldap://<ldap-domain>:389
   activeDirectoryRealm.searchBase = DC=<user-org-level-domain>,DC=<user-
   second-level-domain>,DC=<user-top-level-domain>

   # general settings
   sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
   cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
   securityManager.cacheManager = $cacheManager
   securityManager.sessionManager = $sessionManager
   securityManager.sessionManager.globalSessionTimeout = 86400000

   shiro.loginUrl = /api/login
   ```

3. *Optional*: Zeppelin supports connections to AD over SSL. To force Zeppelin to make an SSL connection to AD, change the value of activeDirectoryRealm.url from ldap to ldaps and specify the AD SSL port; for example:

   ```
   activeDirectoryRealm.url = ldaps://hdp.example.com:636
   ```

   If LDAP is using a self-signed certificate, import the certificate into the truststore of JVM running Zeppelin:

   ```
   echo -n | openssl s_client –connect ldap.example.com:389 | \
       sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/examplecert.
   crt

   keytool –import \
       -keystore $JAVA_HOME/jre/lib/security/cacerts \
       -storepass changeit \
       -noprompt \
       -alias mycert \
       -file /tmp/examplecert.crt
   ```

4. Secure the Websocket channel.

   On an Ambari-managed cluster, navigate to the "Advanced zeppelin-config" section and set zeppelin.anonymous.allowed to false. HDP 2.6: check category

   On a cluster not managed by Ambari, edit the conf/zeppelin-site.xml file. Set zeppelin.anonymous.allowed to false. (If the file does not exist, rename conf/

zeppelin-site.xml.template to conf/zeppelin-site.xml, and then edit zeppelin-site.xml.)

5. *Optional*: If you want to keep clear passwords from appearing in shiro.ini, complete the following steps:

   a. At your OS command line interface, use the Hadoop credential command to create an entry for the Active Directory credential:

   ```
   > hadoop credential create activeDirectoryRealm.systemPassword -provider
    jceks:///etc/zeppelin/conf/credentials.jceks
   Enter password:
   Enter password again:

   activeDirectoryRealm.systemPassword has been successfully created.
   org.apache.hadoop.security.alias.JavaKeyStoreProvider has been updated.
   ```

   b. Using chmod 400, make the credentials.jceks file readable and writable only by the Zeppelin system user.

   c. Add the following line to shiro.ini contents:

   ```
   activeDirectoryRealm.systemPassword -provider jceks://etc/zeppelin/conf/
   credentials.jceks
   ```

6. Restart the Zeppelin server using Ambari or, for a cluster not managed by Ambari, manually restart the Zeppelin server:

   ```
   ./bin/zeppelin-daemon.sh restart
   ```

After successful configuration, Zeppelin requires credentials before allowing users to access the Web UI.

Note: Unless activeDirectoryRealm.principalSuffix is specified, users must fully qualify their account name:

```
ad-username@AD.DOMAIN.COM
```

5.3.2. Configuring Authentication for Production Using LDAP

To use any form of LDAP other than AD, complete the steps in this section.

Note: Zeppelin currently uses LDAP Bind requests to authenticate end users; it does not support the LDAP compare operation.

The following steps describe basic settings. For more information about these and additional settings, see Shiro Settings: Reference and Shiro authentication for Apache Zeppelin.

1. Secure the HTTP channel.

   In the [urls] section of of shiro.ini contents, uncomment the line /** = authc, and comment out the line /** = anon (to disable anonymous access):

```
[urls]
/api/version = anon
#/** = anon
/** = authc
```

Note: The `[urls]` section is processed from top to bottom; earlier statements have precedence. If you have two conflicting lines, the first is honored.

2. In the `[main]` section of `shiro.ini` contents, enable `ldapRealm` and modify the following settings for your operating environment. For clusters managed by Ambari, update `shiro_ini_content`; for non-Ambari clusters, update the `shiro.ini` file.

   Note that there are two types of directory references: those that refer to the LDAP database, and those that refer to user accounts and groups. The domain information can differ between the two.

   ```
   [main]

   # authentication settings
   ldapRealm = org.apache.zeppelin.realm.LdapRealm
   ldapRealm.contextFactory.environment[ldap.searchBase] = DC=<user-second-
   level-domain>,DC=<user-top-level-domain>
   ldapRealm.userDnTemplate = uid={0},OU=<user-account>,DC=<user-second-level-
   domain>,DC=<user-top-level-domain>
   ldapRealm.contextFactory.url = ldap://<ldap-domain>:389
   ldapRealm.contextFactory.authenticationMechanism = simple

   # general settings
   sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
   cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
   securityManager.cacheManager = $cacheManager
   securityManager.sessionManager = $sessionManager
   securityManager.sessionManager.globalSessionTimeout = 86400000

   shiro.loginUrl = /api/login
   ```

3. Optional: Zeppelin supports connections to LDAP over SSL. To force Zeppelin to make an SSL connection to LDAP, change the `contextFactory.url` value from `ldap` to `ldaps` and specify the LDAP SSL port; for example:

   ```
   ldapRealm.contextFactory.url = ldaps://hdp.example.com:636
   ```

   If LDAP is using a self-signed certificate, import the certificate into the truststore of JVM running Zeppelin:

   ```
   echo -n | openssl s_client –connect ldap.example.com:389 | \
       sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /tmp/examplecert.
   crt

   keytool –import \
       -keystore $JAVA_HOME/jre/lib/security/cacerts \
       -storepass changeit \
       -noprompt \
       -alias mycert \
       -file /tmp/examplecert.crt
   ```

4. Secure the Websocket channel.

On an Ambari-managed cluster, set `zeppelin.anonymous.allowed` to `false`.

On a cluster not managed by Ambari, edit the `conf/zeppelin-site.xml` file. Set `zeppelin.anonymous.allowed` to false. (If the file does not exist, rename `conf/zeppelin-site.xml.template` to `conf/zeppelin-site.xml`.)

5. Restart the Zeppelin server using Ambari or, for a cluster not managed by Ambari, manually restart the Zeppelin server:

   ```
   ./bin/zeppelin-daemon.sh restart
   ```

5.4. Enable Access Control for Zeppelin Elements

After configuring authentication, you might want to restrict access control to configure Zeppelin interpreters, access notes, or access data. You can authorize access at three levels within Zeppelin:

- *UI authorization* restricts access to Zeppelin Interpreter, Credential, and Configuration pages based on administrative privileges.
- *Note-level authorization* restricts access to notes based on permissions (owner, reader, or writer) granted to users and groups.
- *Data-level authorization* restricts access to specific data sets.

5.4.1. Enabling Access Control for Interpreter, Configuration, and Credential Settings

By default, any authenticated account can access Zeppelin interpreter, credential, and configuration settings. When access control is enabled, unauthorized users can see the page heading, but no settings.

Prerequisite: Users and groups must be defined on all Zeppelin nodes and in the associated identity store.

To enable access control for the Zeppelin interpreter, credential, or configuration pages, complete the following steps:

1. Define a `[roles]` section in `shiro.ini` contents, and specify permissions for defined groups.

   The following example grants all permissions ("`*`") to users in group `admin`:

   ```
   [roles]
   admin = *
   ```

2. In the `[urls]` section of the `shiro.ini` contents, uncomment the interpreter, configurations, or credential line(s) to enable access to the interpreter, configuration, or credential page(s), respectively. (If the `[urls]` section is not defined, add the section. Include the three `/api` lines listed in the following example.)

The following example specifies access to interpreter, configurations, and credential settings for role "admin":

```
[urls]
/api/version = anon
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
#/** = anon
/** = authc
```

To add more roles, separate role identifiers with commas inside the square brackets.

Note: The sequence of lines in the `[urls]` section is important. The `/api/version` line must be the first line in the `[urls]` section:

```
/api/version = anon
```

Next, specify the three `/api` lines in any order:

```
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
```

The `authc` line must be last in the `[urls]` section:

```
/** = authc
```

3. Map the roles to LDAP or Active Directory (AD) groups. The following is an example of the `shiro.ini` settings for Active Directory (before pasting this configuration in your Zeppelin configuration, update the Active Directory details to match your actual configuration settings).

```
# Sample LDAP configuration, for Active Directory user Authentication,
 currently tested for single Realm
[main]
ldapRealm=org.apache.zeppelin.realm.LdapRealm
ldapRealm.contextFactory.systemUsername=cn=ldap-reader,ou=ServiceUsers,dc=
lab,dc=hortonworks,dc=net
ldapRealm.contextFactory.systemPassword=SomePassw0rd
ldapRealm.contextFactory.authenticationMechanism=simple
ldapRealm.contextFactory.url=ldap://ad.somedomain.net:389
# Ability to set ldap paging Size if needed; default is 100
ldapRealm.pagingSize=200
ldapRealm.authorizationEnabled=true
ldapRealm.searchBase=OU=CorpUsers,DC=lab,DC=hortonworks,DC=net
ldapRealm.userSearchBase=OU=CorpUsers,DC=lab,DC=hortonworks,DC=net
ldapRealm.groupSearchBase=OU=CorpUsers,DC=lab,DC=hortonworks,DC=net
ldapRealm.userObjectClass=person
ldapRealm.groupObjectClass=group
ldapRealm.userSearchAttributeName = sAMAccountName
# Set search scopes for user and group. Values: subtree (default), onelevel,
 object
ldapRealm.userSearchScope = subtree
ldapRealm.groupSearchScope = subtree
ldapRealm.userSearchFilter=(&(objectclass=person)(sAMAccountName={0}))
ldapRealm.memberAttribute=member
# Format to parse & search group member values in 'memberAttribute'
```

```
ldapRealm.memberAttributeValueTemplate=CN={0},OU=CorpUsers,DC=lab,DC=
hortonworks,DC=net
# No need to give userDnTemplate if memberAttributeValueTemplate is provided
#ldapRealm.userDnTemplate=
# Map from physical AD groups to logical application roles
ldapRealm.rolesByGroup = "hadoop-admins":admin_role,"hadoop-
users":hadoop_users_role
# Force usernames returned from ldap to lowercase, useful for AD
ldapRealm.userLowerCase = true

# Enable support for nested groups using the LDAP_MATCHING_RULE_IN_CHAIN
 operator
ldapRealm.groupSearchEnableMatchingRuleInChain = true

sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
### If caching of user is required then uncomment below lines
cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
securityManager.cacheManager = $cacheManager

securityManager.sessionManager = $sessionManager
securityManager.realms = $ldapRealm
# 86,400,000 milliseconds = 24 hour
securityManager.sessionManager.globalSessionTimeout = 86400000
shiro.loginUrl = /api/login

[urls]
# This section is used for url-based security.
# You can secure interpreter, configuration and credential information by
 urls. Comment or uncomment the below urls that you want to hide.
# anon means the access is anonymous.
# authc means Form based Auth Security
# To enfore security, comment the line below and uncomment the next one
#/api/version = anon
/api/interpreter/** = authc, roles[admin_role,hadoop_users_role]
/api/configurations/** = authc, roles[admin_role]
/api/credential/** = authc, roles[admin_role,hadoop_users_role]
#/** = anon
/** = authc
```

Additional information:

- `ldapRealm.rolesByGroup = "hadoop-admins":admin_role,"hadoop-users":hadoop_users_role`

  This line maps the AD groups "hadoop-admins" and "hadoop-users" to custom roles which can be used in the [urls] section to control access to various Zeppelin users. Note that the short group names are to be used rather than fully qualified names such as "cn=hadoop-admins,OU=CorpUsers,DC=lab,DC=hortonworks,DC=net". The role names can be set to any name but the names should match those used in the [urls] section.

- `ldapRealm.groupSearchEnableMatchingRuleInChain = true`

  A very powerful option to search all of the groups that a given user is member of in a single query. An LDAP search query with this option traverses the LDAP group hierarchy and finds all of the groups. This is especially useful for nested groups. More information can be found here. Caution: this option can cause performance overhead (slow to log in, etc.) if the LDAP hierarchy is not optimally configured.

- `ldapRealm.userSearchFilter=(&(objectclass=person)(sAMAccountName={0}))`

  Use this search filter to limit the scope of user results when looking for a user's Distinguished Name (DN). This is used only if `userSearchBase` and `userSearchAttributeName` are defined. If these two are not defined, `userDnTemplate` is used to look for a user's DN.

4. When unauthorized users attempt to access the interpreter, configurations, or credential page, they see the page heading, but not the settings.

5.4.2. Enabling Access Control for Notebooks

The next optional security step is to add access control for Zeppelin notes, granting permissions to specific users and groups. There are two main steps to this process: defining the `searchBase` property in the Zeppelin Shiro configuration, and then specifying permissions.

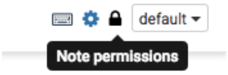
To restrict notebook access to authorized users:

1. In Zeppelin configuration settings, the Zeppelin administrator should specify `activeDirectoryRealm.searchBase` or `ldapRealm.searchBase`, depending on whether Zeppelin uses AD or LDAP for authentication. The value of `searchBase` controls where Zeppelin looks for users and groups.

   For more information, refer to Shiro Settings: Reference. For an example, see Configure Authentication for Authentication: LDAP and Active Directory.

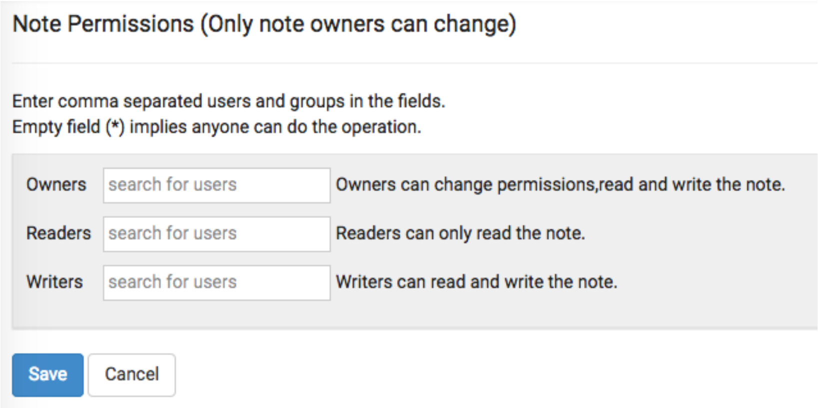
2. The owner of the notebook should navigate to the note and complete the following steps:

   a. Click the lock icon on the notebook:

   b. Zeppelin presents a popup menu. Enter the user and groups that should have access to the note. To search for an account, start typing the name.

   Note: If you are using Shiro as the identity store, users should be listed in the `[user]` section. If you are using AD or LDAP users and groups should be stored in the realm associated with your Shiro configuration.



Note Permissions (Only note owners can change)

Enter comma separated users and groups in the fields.
Empty field (*) implies anyone can do the operation.

Owners	search for users	Owners can change permissions,read and write the note.
Readers	search for users	Readers can only read the note.
Writers	search for users	Writers can read and write the note.

For more information, see Apache Zeppelin Notebook Authorization.

5.4.3. Enabling Access Control for Data

Access control for data brought into Zeppelin depends on the underlying data source:

- To configure access control for Spark data, Zeppelin must be running as an end user ("identity propagation"). Zeppelin implements access control using Livy. When identity propagation is enabled via Livy, data access is controlled by the type of data source being accessed. For example, when you access HDFS data, access is controlled by HDFS permissions.
- To configure access control for Hive data, use the JDBC interpreter.
- To configure access control for the Spark shell, define permissions for end users running the shell.

5.5. Configure Zeppelin JDBC Password Security Using JCEKS

Apache Zeppelin supports JDBC intepreter passwords stored in JCEKS, to avoid specifying passwords in clear text. This feature prevents users from reading clear text passwords from the interpreter JSON file using the shell or other tools.

Use the following steps to configure Zeppelin JCEKS password security on an Ambari cluster.

1. Create a keystore file using the hadoop credential command line command, with hadoop commons in the classpath:

   ```
   hadoop credential create jdbc.password -provider jceks://file/user/zeppelin/
   conf/zeppelin.jceks
   ```

2. In the Zeppelin UI, click the user settings menu at the top right, then select **Interpreter**. On the Interpreters page, set the following jdbc properties:

name	value
default.driver	org.postgresql.Driver

name	value
default.jceks.credentialKey	jdbc.password
default.jceks.file	jceks://file/tmp/zeppelin.jceks
default.url	jdbc:postgresql://localhost:5432/
default.user	rk-user

5.6. Configure SSL for Zeppelin

To configure the Zeppelin UI for access over SSL:

1. In Ambari, access the "Advanced zeppelin-config" section of the Zeppelin configuration settings:

2. Set `zeppelin.ssl` to true.

3. Configure key manager and key store settings with the correct values for your system:

   a. Set `zeppelin.ssl.key.manager.password` to the password associated with the key manager.

   b. Set `zeppelin.ssl.keystore.password` to the password associated with the key store.

   c. Set `zeppelin.ssl.keystore.path` to the path associated with the key store.

   d. Set `zeppelin.ssl.keystore.type` to the type of key store configured on the cluster (for example, JKS).

4. If you wish to use client-side certificate authentication, enable client-side authentication and configure the associated trust store settings:

   a. Set `zeppelin.ssl.cient.auth` to `true`

   b. Set `zeppelin.ssl.truststore.path` to the path associated with your trust store.

   c. Set `zeppelin.ssl.truststore.password` to the password associated with your trust store.

   d. Set `zeppelin.ssl.truststore.type` to the type of trust store configured on the cluster (for example, JKS).

5. Check to make sure that all settings are valid.

6. Connect using HTTPS.

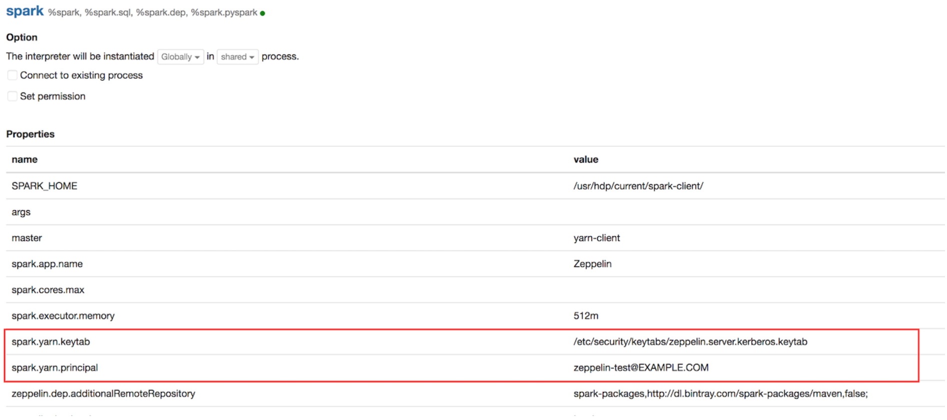
Note: When SSL is enabled for Zeppelin, the Safari browser requires a Certificate Authority-signed certificate to access the Zeppelin UI.

5.7. Configure Zeppelin for a Kerberos-Enabled Cluster

The Zeppelin daemon needs a Kerberos account and keytab to run in a Kerberized cluster.

- When you enable Kerberos on an Ambari-managed cluster, Ambari configures Kerberos for Zeppelin and automatically creates a Kerberos account and keytab for it. For more information, see Configuring Ambari and Hadoop for Kerberos.
- If your cluster is not managed with Ambari and you plan to enable Kerberos for the Zeppelin server, see Creating Service Principals and Keytab Files for HDP in the Hadoop Security Guide.

After configuring Kerberos for Zeppelin in Ambari, you can find all related settings on the Zeppelin Interpreter settings page, as shown in the following image for the `%spark` interpreter. If you configured Kerberos from Ambari, no further action is needed. Changes in values for keytabs and principals are managed by Ambari, and if Kerberos is disabled, Ambari deletes keytab and principal values.



spark %spark, %spark.sql, %spark.dep, %spark.pyspark ●

Option

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

□ Connect to existing process

□ Set permission

Properties

name	value
SPARK_HOME	/usr/hdp/current/spark-client/
args	
master	yarn-client
spark.app.name	Zeppelin
spark.cores.max	
spark.executor.memory	512m
spark.yarn.keytab	/etc/security/keytabs/zeppelin.server.kerberos.keytab
spark.yarn.principal	zeppelin-test@EXAMPLE.COM
zeppelin.dep.additionalRemoteRepository	spark-packages,http://dl.bintray.com/spark-packages/maven,false;

For clusters not managed by Ambari, note that every interpreter that supports Kerberos has two configuration properties: keytab and principal. In addition, the Shell interpreter (`%sh`) has a property for specifying authentication method: `zeppelin.shell.auth.type`. Set authentication method to KERBEROS for a Kerberos-enabled cluster; otherwise the value should be empty.

The following table lists properties used for keytabs and principals, for each associated interpreter.

Interpreter	Keytab Property	Principal Property
%jdbc	zeppelin.jdbc.keytab.location	zeppelin.jdbc.principal
%livy	zeppelin.livy.keytab	zeppelin.livy.principal
%sh	zeppelin.shell.keytab.location	zeppelin.shell.principal
%spark	spark.yarn.keytab	spark.yarn.principal

5.8. Shiro Settings: Reference

This section lists more information about Shiro settings described in the authentication and authorization configuration subsections, plus a few more settings that might be useful.

5.8.1. Active Directory Settings

Active Directory (AD) stores users and groups in a hierarchical tree structure, built from containers including the organizational unit (ou), organization (o), and domain controller (dc). The path to each entry is a Distinguished Name (DN) that uniquely identifies a user or group.

User and group names typically have attributes such as a common name (cn) or unique ID (uid).

Specify the DN as a string, for example `cn=admin,dc=example,dc=com`. White space is ignored.

Important

If you upgrade from HDP 2.5 (Zeppelin 0.6) to HDP 2.6 (Zeppelin 0.7), note that the Active Directory group realm class name changed from `org.apache.zeppelin.server.ActiveDirectoryGroupRealm` to `org.apache.zeppelin.realm.ActiveDirectoryGroupRealm` in Zeppelin 0.7. For information about changing this setting manually, see Configuring and Upgrading Apache Zeppelin in the *Command Line Upgrade Guide*.

`activeDirectoryRealm`

specifies the class name to use for AD authentication. You should set this to `org.apache.zeppelin.realm.ActiveDirectoryGroupRealm`. For more information, see Apache Shiro Realms.

`activeDirectoryRealm.url`

specifies the host and port where Active Directory is set up. For more information, see Apache Shiro Realms.

If the protocol element is specified as `ldap`, SSL is not used. If the protocol is specified as `ldaps`, access is over SSL.

Note: If Active Directory uses a self-signed certificate, import the certificate into the truststore of the JVM running Zeppelin; for example:

```
echo -n | openssl s_client –connect ldap.
example.com:389 | \
    sed -ne '/-BEGIN CERTIFICATE-/,/-END
 CERTIFICATE-/p' > /tmp/examplecert.crt

keytool –import \
    -keystore $JAVA_HOME/jre/lib/security/
cacerts \
    -storepass changeit \
    -noprompt \
    -alias mycert \
    -file /tmp/examplecert.crt
```

`activeDirectoryRealm.principalSuffix`

simplifies the logon information that users must use to log in. Otherwise, AD requires a username fully qualified with domain information. For example, if a fully-qualified user account is `user@hdpqa.example.com`, you can specify a shorter suffix such as `user@hdpqa`.

```
activeDirectoryRealm.principalSuffix = @<user-
org-level-domain>
```

`activeDirectoryRealm.searchBase`

defines the base distinguished name from which the directory search starts. A distinguished name defines each entry; "dc" entries define a hierarchical directory tree.

`activeDirectoryRealm.systemUsername`, `activeDirectoryRealm.systemPassword`

define the username and password that Zeppelin uses to connect to Active Directory when it searches for users and groups. These two settings are used for controlling access to UI features, not for authentication. The Bind method does not require a valid user password.

Example: `activeDirectoryRealm.systemPassword = passwordA`

`activeDirectoryRealm.groupRolesMap`

a comma-separated list that maps groups to roles. These settings are used by Zeppelin to restrict UI features to specific AD groups. The following example maps group `hdpdv_admin` at `hdp3.example.com` to the "admin" role:

```
CN=hdpdv_admin,DC=hdp3,DC=example,DC=com:admin
```

`activeDirectoryRealm.authorizationCachingEnabled`

specifies whether to use caching to improve performance. To enable caching, set this property to `true`.

5.8.2. LDAP Settings

LDAP stores users and groups in a hierarchical tree structure, built from containers including the organizational unit (ou), organization (o), and domain controller (dc). The path to each entry is a Distinguished Name (DN) that uniquely identifies a user or group.

User and group names typically have attributes such as a common name (cn) or unique ID (uid).

Specify the DN as a string, for example `cn=admin,dc=example,dc=com`. White space is ignored.

Zeppelin LDAP authentication uses templates for user DNs. SeeLDAP Realm Settings for options you can set for an LDAP realm.

`ldapRealm`

specifies the class name to use for LDAP authentication. You should set this to `org.apache.zeppelin.realm.LdapRealm` unless you are familiar with LDAP and prefer to use `org.apache.shiro.realm.ldap.JndiLdapRealm`. For more information, see Apache Shiro Realms.

`ldapRealm.contextFactory.environment[ldap.searchBase]`

defines the base distinguished name from which the LDAP search starts. Shiro searches for `userDnTemplate` at this address.

If the protocol is specified as `ldap`, SSL is not used. If the protocol is specified as `ldaps`, access is over SSL.

`ldapRealm.userDnTemplate`

specifies the search location where the user is to be found. Shiro replaces `{0}` with the username acquired from the Zeppelin UI. Zeppelin uses User DN templates to configure associated realms.

`ldapRealm.contextFactory.url`

specifies the host and port on which LDAP is running.

If the protocol element is specified as `ldap`, SSL will not be used. If the protocol is specified as `ldaps`, access will be over SSL.

Note: If LDAP is using a self-signed certificate, import the certificate into the truststore of JVM running Zeppelin; for example:

```
echo -n | openssl s_client –connect ldap.
example.com:389 | \
    sed -ne '/-BEGIN CERTIFICATE-/,/-END
 CERTIFICATE-/p' > /tmp/examplecert.crt

keytool –import \
    -keystore $JAVA_HOME/jre/lib/security/
cacerts \
    -storepass changeit \
    -noprompt \
    -alias mycert \
    -file /tmp/examplecert.crt
```

`ldapRealm.contextFactory.systemUsername,`
`ldapRealm.contextFactory.systemPassword`

defines the username and password that Zeppelin uses to connect to LDAP, to search for users and groups. These two settings are used for controlling access to UI features, not for authentication. The Bind method does not require a valid user password.

Examples:

```
ldapRealm.contextFactory.systemUsername=uid=
guest,ou=people,dc=hadoop,dc=apache,dc=org
```

```
ldapRealm.contextFactory.systemPassword=
somePassword
```

`ldapRealm.authorizationCachingEnabled`

Specifies whether to use caching to improve performance. To enable caching, set this property to `true`.

5.8.3. General Settings

`securityManager.sessionManager.globalSessionTimeout`

Specifies how long to wait (in milliseconds) before logging out a user, if they are logged in and are not moving the cursor.

The default is 86,400,000 milliseconds, which equals 24 hours.

5.9. shiro.ini Example

The following example shows a minimum set of `shiro.ini` settings for authentication and access control, for a Zeppelin deployment that uses Active Directory.

Prerequisite: corresponding account information is configured in Active Directory (at `adhost.field.hortonworks.com`, in this example) and on Zeppelin nodes.

```
[main]
# AD authentication settings
activeDirectoryRealm = org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
activeDirectoryRealm.url = ldap://adhost.org.hortonworks.com:389
activeDirectoryRealm.searchBase = DC=org,DC=hortonworks,DC=com

# general settings
sessionManager = org.apache.shiro.web.session.mgt.DefaultWebSessionManager
cacheManager = org.apache.shiro.cache.MemoryConstrainedCacheManager
securityManager.cacheManager = $cacheManager
securityManager.sessionManager = $sessionManager
securityManager.sessionManager.globalSessionTimeout = 86400000
shiro.loginUrl = /api/login

[roles]
admin = *

[urls]
# authentication method and access control filters
/api/version = anon
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
#/** = anon
/** = authc
```

6. Stopping the Zeppelin Server and Livy Server

To stop the Zeppelin server on a cluster managed by Ambari, use the Ambari Web UI.

To stop the Zeppelin server on a cluster that is not managed by Ambari, issue the following command from user `zeppelin`:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh stop
```

To stop the Livy server, issue the following command from user `livy`:

```
/usr/hdp/<hdp_version>/livy/bin/livy-server stop
```

For example:

```
/usr/hdp/2.6.0.0-598/livy/bin
```

7. Enabling HDFS Storage for Zeppelin Notebooks and Configuration in HDP-2.6.3+

7.1. Overview

HDP-2.6.3 introduced support for HDFS storage for Apache Zeppelin notebooks and configuration files. In previous versions, notebooks and configuration files were stored on the local disk of the Zeppelin server.

When upgrading to HDP-2.6.3 and higher versions, there are two options for configuring Zeppelin notebook and configuration file storage:

- **Use HDFS storage (recommended)** – Zeppelin notebooks and configuration files must be copied to the new HDFS storage location before upgrading. Additional upgrade and post-upgrade steps must also be performed, as described in the following section.

- **Use local storage** – Perform upgrade and post-upgrade steps to enable local storage.

Enabling HDFS storage makes future upgrades much easier, and also sets up the first step toward enabling Zeppelin High Availability. Therefore it is recommended that you enable HDFS for Zeppelin notebooks and configuration files when upgrading to HDP 2.6.3+ from earlier versions of HDP.

Note

Currently HDFS and local storage are the only supported notebook storage mechanisms in HDP-2.6.3+. Currently VFSNotebookRepo is the only supported local storage option.

7.2. Enable HDFS Storage when Upgrading to HDP-2.6.3+

Perform the following steps to enable HDFS storage when upgrading to HDP 2.6.3+ from earlier versions of HDP.

1. Before upgrading Zeppelin, perform the following steps as the Zeppelin service user.

   a. Create the `/user/zeppelin/conf` and `/user/zeppelin/notebook` directories in HDFS.

```
hdfs dfs -ls /user/zeppelin
drwxr-xr-x   - zeppelin hdfs          0 2018-01-20 04:17 /user/zeppelin/
conf
drwxr-xr-x   - zeppelin hdfs          0 2018-01-20 03:40 /user/zeppelin/
notebook
```

   b. Copy all notebooks from the local Zeppelin server (for example, `/usr/hdp/2.5.3.0-37/zeppelin/notebook/`) to the `/user/zeppelin/notebook` directory in HDFS.

      ```
      hdfs dfs -ls /user/zeppelin/notebook
      drwxr-xr-x   - zeppelin hdfs          0 2018-01-19 01:40 /user/zeppelin/
      notebook/2A94M5J1Z
      drwxr-xr-x   - zeppelin hdfs          0 2018-01-19 01:40 /user/zeppelin/
      notebook/2BWJFTXKJ
      ```

   c. Copy the `interpreter.json` and `notebook-authorization.json` files from the local Zeppelin service configuration directory (`/etc/zeppelin/conf`) to the `/user/zeppelin/conf` directory in HDFS.

      ```
      hdfs dfs -ls /user/zeppelin/conf
      -rw-r--r--   3 zeppelin hdfs     284091 2018-01-22 23:28 /user/zeppelin/
      conf/interpreter.json
      -rw-r--r--   3 zeppelin hdfs     123849 2018-01-22 23:29 /user/zeppelin/
      conf/notebook-authorization.json
      ```

2. Upgrade Ambari.

3. Upgrade HDP and Zeppelin. During the upgrade, verify that the following configuration settings are present in Ambari for Zeppelin.

   ```
   zeppelin.notebook.storage = org.apache.zeppelin.notebook.repo.
   FileSystemNotebookRepo
   zeppelin.config.fs.dir = conf
   ```

   If necessary, add or update these configuration settings as shown above.

4. After the upgrade is complete:

   a. Log on to the Zeppelin server and verify that the following properties exist in the `/etc/zeppelin/conf/zeppelin-site.xml` file. The actual value for the keytab file and principal name may be different for your cluster.

      ```
      <property>
        <name>zeppelin.server.kerberos.keytab</name>
        <value>/etc/security/keytabs/zeppelin.server.kerberos.keytab</value>
      </property>
      <property>
        <name>zeppelin.server.kerberos.principal</name>
        <value>zeppelin@EXAMPLE.COM</value>
      </property>
      ```

   b. Check the Zeppelin Interpreter page to see if any interpreter (e.g. the Livy interpreter) is duplicated. This may happen in some cases. If duplicate interpreter entries are found, perform the following steps:

      i. Backup and delete the interpreter.json file from HDFS (`/user/zeppelin/conf/interpreter.json`) and from the local Zeppelin server.

      ii. Restart the Zeppelin service.

      iii. Verify that the duplicate entries no longer exist.

iv. If any custom interpreter settings were present before the upgrade, add them again via the Zeppelin interpreter UI page.

c. Verify that your existing notebooks are available on Zeppelin.

Note

When an existing notebook is opened for the first time after the upgrade, it may ask you to save the interpreters associated with the notebook.

7.3. Use Local Storage when Upgrading to HDP-2.6.3+

Perform the following steps to use local notebook storage when upgrading to HDP 2.6.3+ from earlier versions of HDP.

1. Upgrade Ambari.

2. Upgrade HDP and Zeppelin. During the upgrade, verify that the following configuration settings are present in Ambari for Zeppelin.

   ```
   zeppelin.notebook.storage = org.apache.zeppelin.notebook.repo.
   VFSNotebookRepo
   zeppelin.config.fs.dir = file:///etc/zeppelin/conf
   ```

   If necessary, add or update these configuration settings as shown above.

3. After the upgrade is complete:

   a. Copy your notebooks and the `notebook-authorization.json` file from the previous Zeppelin installation directory to the new installation directory on the Zeppelin server machine.

   b. Verify that your existing notebooks are available on Zeppelin.

Note

When an existing notebook is opened for the first time after the upgrade, it may ask you to save the interpreters associated with the notebook.