Cloudera Runtime 7.3.1

Apache Zeppelin

Date published: 2020-07-28
Date modified: 2024-12-10

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Zeppelin Overview

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

Important:

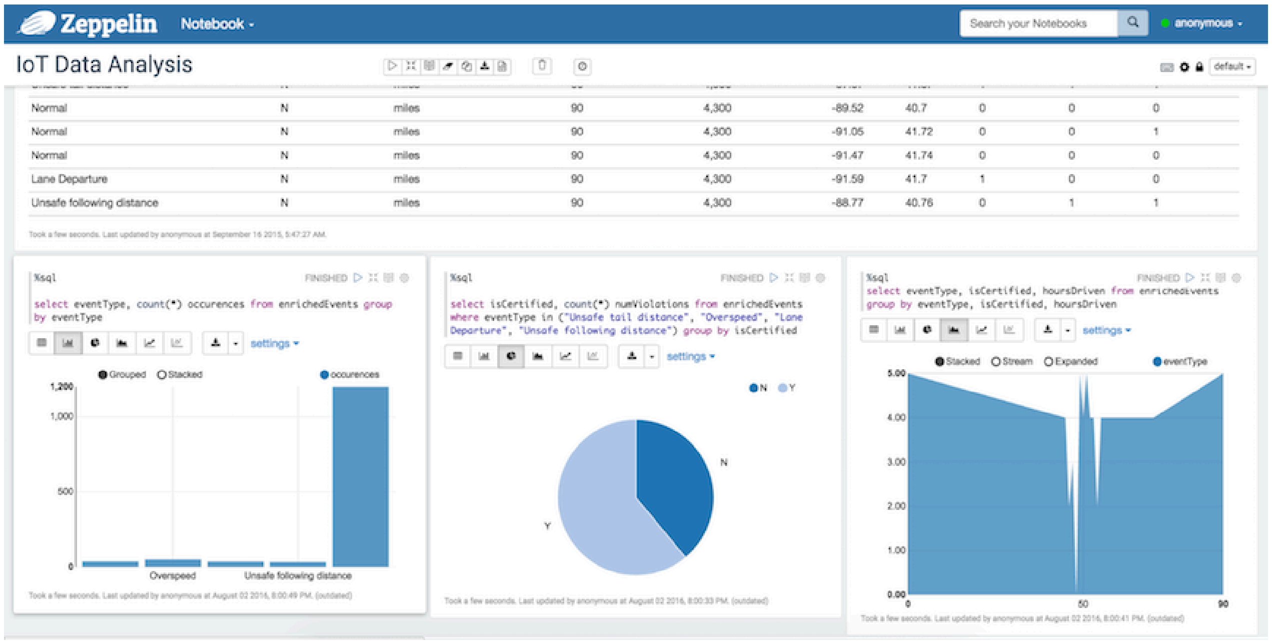
Zeppelin has been deprecated in Cloudera Runtime 7.2.187.1.9 and is no longer supported by Cloudera.

You can reinstall Zeppelin and migrate your previously used Zeppeling notebooks at your own risk.

Refer to *Reinstall Apache Zeppelin in 7.3.1* for more information on how to reinstall Zeppelin in Cloudera Private Cloud 7.3.1.

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.



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
```

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 (outdated)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
```

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 (outdated)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
```

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 (outdated)

Installing Apache Zeppelin

How to install Apache Zeppelin.

Reinstall Apache Zeppelin in 7.3.1

How to reinstall Apache Zeppelin in Cloudera Private Cloud 7.3.1.

About this task

Note: Customers are advised to take a backup of their existing notebooks by logging into their respective notebook repositories and creating a copy of the same, interpreter config (interpreter.json file) and safety variables added in the CM UI as configuration (zeppelin-site.xml and Zeppelin-env.sh) before Upgrade.

Important:

Cloudera Manager changes for Zeppelin in 7.3.1

Zeppelin has been deprecated in Cloudera Runtime 7.1.9 and is no longer supported by Cloudera.

1. Fresh Cluster Installs
    - Zeppelin doesn't show up in the Install wizard.
    - Zeppeling doesn't show up in the list of available services of the Add-service wizard of Cloudera Manager.
2. Upgraded Clusters
    a. During the pre-upgrade check, Cloudera Manager will show a warning that Zeppelin is going to be disabled when upgrading to 7.3.1. The upgrade will not proceed until Zeppelin is stopped and deleted from the cluster.
    b. Once the Zeppelin service is stopped and deleted, cluster upgrade can be continued. The upgraded cluster will no longer contain Zeppelin.

You can reinstall the Zeppelin service as an external CSD as shown below, but Cloudera will not provide support for Zeppelin in 7.3.1.

For more information on building and using external CSDs, see:

- *Cloudera Manager Add-on Services*
- *Cloudera Manager Extensions | Cloudera GitHub Wiki*
- *Custom Service Descriptors | Cloudera GitHub Wiki*
- *Cloudera CSD repository | GitHub*

Procedure

1. Download the CSD Build Code.
    a) Clone the repository containing the code required to build CSDs:

    ```
    git clone https://github.com/cloudera/cm_csds.git
    ```

2. Copy the interpreter config file (interpreter.json) for Zeppelin to the source of the Zeppelin CSD folder cm_csds/ZEPPELIN/src/aux/ inside the downloaded cm_csds repository.
3. Build the CSD JAR File.
    a) Navigate to the cm_csds directory and build the JAR file for Zeppelin:

    ```
    cd cm_csds
    mvn clean install
    ```

    b) After the build completes, you'll find the CSD JAR file in the target directory:

    ```
    ls -lrt ZEPPELIN/target/ZEPPELIN*
    ```

4. Deploy the CSD JAR File to Cloudera Manager.

a) Copy the CSD JAR file to the Cloudera Manager server:

```
scp ZEPPELIN/target/ZEPPELIN-7.3.1.jar [*** USERNAME ***]@[*** HOST
 ***]:/opt/cloudera/csd/
```

b) SSH into the Cloudera Manager server:

```
ssh [*** USERNAME ***]@[*** HOST ***]
```

c) Navigate to the CSD directory:

```
cd /opt/cloudera/csd/
```

d) Set the correct permissions and ownership for the JAR file:

```
sudo chmod 664 ZEPPELIN-7.3.1.jar
sudo chown cloudera-scm:cloudera-scm ZEPPELIN-7.3.1.jar
```

e) Restart the Cloudera Manager server:

```
sudo service cloudera-scm-server restart
```

5. Configure and Add Zeppelin Service.

a) Open **Cloudera Manager Admin Console**.
b) Add the Zeppelin service in the Cloudera Manager Home page: CM HomeStatus3 dotsAdd service
c) Select Zeppelin.
d) Follow the on-screen instructions to add the Zeppelin service to your cluster.
e) Configure authentication and create users as needed by your authentication provider.
f) Start Zeppelin.
g) Open Zeppelin in the browser.
h) Test by running any paragraph to confirm it is working as expected.
i) Replace the notebook folder with the earlier backed up notebooks.
j) Add the earlier backed up safety variables (zeppelin-site.xml and zeppelin-env.sh) in Cloudera ManagerZeppelinConfigs.
k) Restart Zeppelin.

Results

Zeppelin is available in your cluster and you are able to run the older notebooks.

Related Information

Cloudera Manager Extensions | Cloudera GitHub Wiki
Custom Service Descriptors | Cloudera GitHub Wiki
Cloudera CSD repository | GitHub
Cloudera Manager Add-on Services

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

Overview

When upgrading to HDP-2.6.3 and higher versions, additional configuration steps are required to enable HDFS storage for Apache Zeppelin notebooks.

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.

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.

Procedure

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.FileSystem
   NotebookRepo
   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:

      1. Backup and delete the interpreter.json file from HDFS (/user/zeppelin/conf/interpreter.json) and from the local Zeppelin server.
      2. Restart the Zeppelin service.
      3. Verify that the duplicate entries no longer exist.
      4. 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.

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.

Procedure

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.VFSNoteboo
   kRepo
   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.

Configuring Apache Zeppelin

How to configure Apache Zeppelin.

Introduction

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

Clusters created in CDP have security enabled by default. Disabling security on a cluster in CDP is not supported.

Important: Zeppelin will be deprecated in Cloudera Runtime 7.2.18. For more information, see *Deprecation Notices in Cloudera Runtime 7.2.18*.

What to do next

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.

Configuring Zeppelin caching

Improve cache management and server response for Zeppelin.

About this task

There are two configuration properties in Zeppelin to control the HTTP response headers, allowing for improved cache management and server response behavior:

- zeppelin.server.response.header.cache-control - default value: no-cache, no-store, must-revalidate, no-transform
- zeppelin.server.response.header.pragma - default value: no-cache

Procedure

1. Navigate to Cloudera ManagerZeppelinConfiguration.
2. Add zeppelin.server.response.header.cache-control and zeppelin.server.response.header.pragma with your desired values under Zeppelin Server Advanced Configuration Snippet (Safety Valve) for zeppelin-conf/zeppelin-site.xml.
3. Restart the Zeppelin service.

Configuring Livy

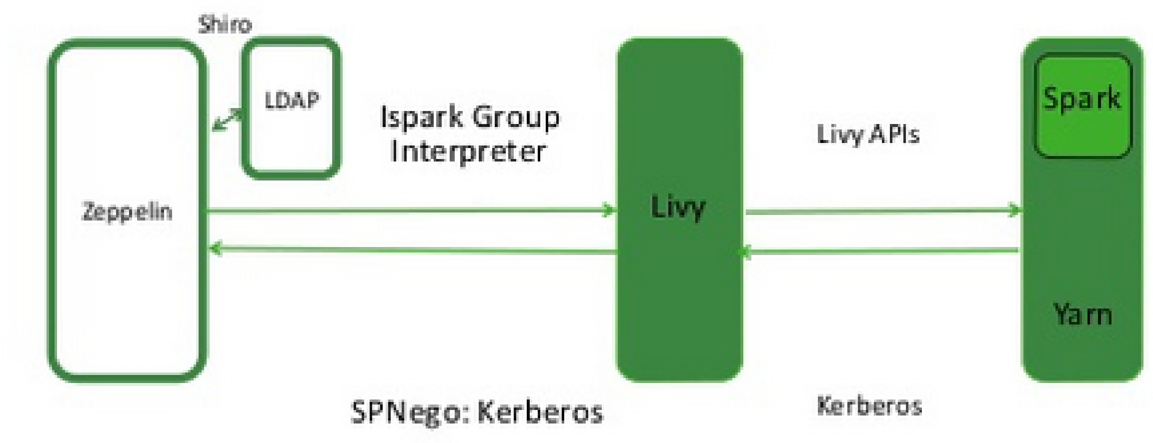
About this task

This section describes how to configure Livy in CDP.

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.

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



The following sections describe several optional configuration steps.

Configure Livy user access control

You can use the livy.server.access-control.enabled property to configure Livy user access.

When this property is set to false, only the session owner and the superuser can access (both view and modify) a given session. Users cannot access sessions that belong to other users. ACLs are disabled, and any user can send any request to Livy.

When this property is set to true, ACLs are enabled, and the following properties are used to control user access:

- livy.server.access-control.allowed-users – A comma-separated list of users who are allowed to access Livy.
- livy.server.access-control.view-users – A comma-separated list of users with permission to view other users' infomation, such as submitted session state and statement results.
- livy.server.access-control.modify-users – A comma-separated list of users with permission to modify the sessions of other users, such as submitting statements and deleting the session.

Restart the Livy interpreter after changing settings

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, then click restart.

Verify that the Livy server is running

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/
```

Livy high availability support

Livy supports high availability. If there is more than one Livy server in the CDP cluster, high availability is automatically enabled by Cloudera Manager by setting the `livy.server.recovery.mode` property to `ha` and by including the list of Zookeeper servers in the Livy configuration.

The new high availability mode runs multiple instances of Livy in a cluster. Both the active and passive instances continuously run and the passive instances take the active role when required. There is only one active server at a time, and the remaining instances are passive. Passive instances only redirect requests to the active server using HTTP 307.

Limitations

- **JDBC connection**: The JDBC connections are not redirected. More details about this limitation are available below.
- **Load balancing**: The active-passive high availability model does not provide additional parallelism or capacity. Interactive sessions are only handled by the active server, adding more servers does not distribute load across servers.

Using Livy without Knox gateway

If you are not using Livy through Knox gateway, clients must follow HTTP redirects and resend authentication. The logic the clients can use is to make a list of the Livy servers by obtaining them from the Cloudera Manager API and if any of the servers do not respond, the clients should retry sending the request to another server in the list. For example, in the case of cURL, the `--location-trusted` flag has to be specified to follow redirects and resend authentication.

Livy high availability and JDBC connection

Livy provides high availability by active-passive setup. Only the active Livy server node can accept JDBC connections, and the passive nodes reject connection attempts. Therefore, if a client wants to connect using JDBC, it has to iterate through all servers and check which one accepts connections. If the active server goes down, the connection is broken and another node takes over the active role. This behavior is the same for both HTTP and binary mode connections.

Configure User Impersonation for Access to Hive

This section describes how to configure Apache Zeppelin user impersonation for Apache Hive.

About this task

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

Kerberos-enabled Cluster

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. 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/;service
   DiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2
   ```
5. Add a hive.proxy.user.property property and set its value tohive.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" in this guide.

Configure User Impersonation for Access to Phoenix

This section describes how to configure Apache Zeppelin user impersonation for Apache Phoenix.

About this task

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

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

Procedure

1. In the 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.

Enabling Access Control for Zeppelin Elements

This section describes how to restrict access to specific Apache Zeppelin elements.

After configuring authentication, you may want to restrict access to Zeppelin notes and data, and also set restrictions on what users and groups can configure Zeppelin interpreters. 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.

Enable Access Control for Interpreter, Configuration, and Credential Settings

This section describes how to restrict access to Apache Zeppelin interpreter, credential, and configuration settings.

About this task

By default, any authenticated account can access the Zeppelin interpreter, credential, and configuration settings. When access control is enabled, unauthorized users can see the page heading, but not the settings.

Before you begin

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

Procedure

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 [url s] 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, c
urrently 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), onel
evel, 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=horto
nworks,DC=net
# No need to give userDnTemplate if memberAttributeValueTemplate is provid
ed
#ldapRealm.userDnTemplate=
# Map from physical AD groups to logical application roles
ldapRealm.rolesByGroup = "hadoop-admins":admin_role,"hadoop-users":hado
op_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 can see the page heading, but not the settings.

Enable Access Control for Notebooks

This section describes how to restrict access to Apache Zeppelin notebooks by granting permissions to specific users and groups.

About this task

There are two main steps in this process: defining the searchBase property in the Zeppelin Shiro configuration, and then specifying permissions.

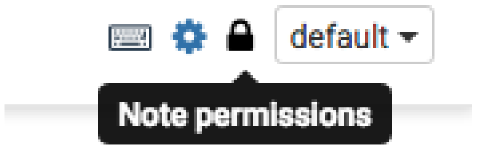
Procedure

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 sear chBase controls where Zeppelin looks for users and groups.

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

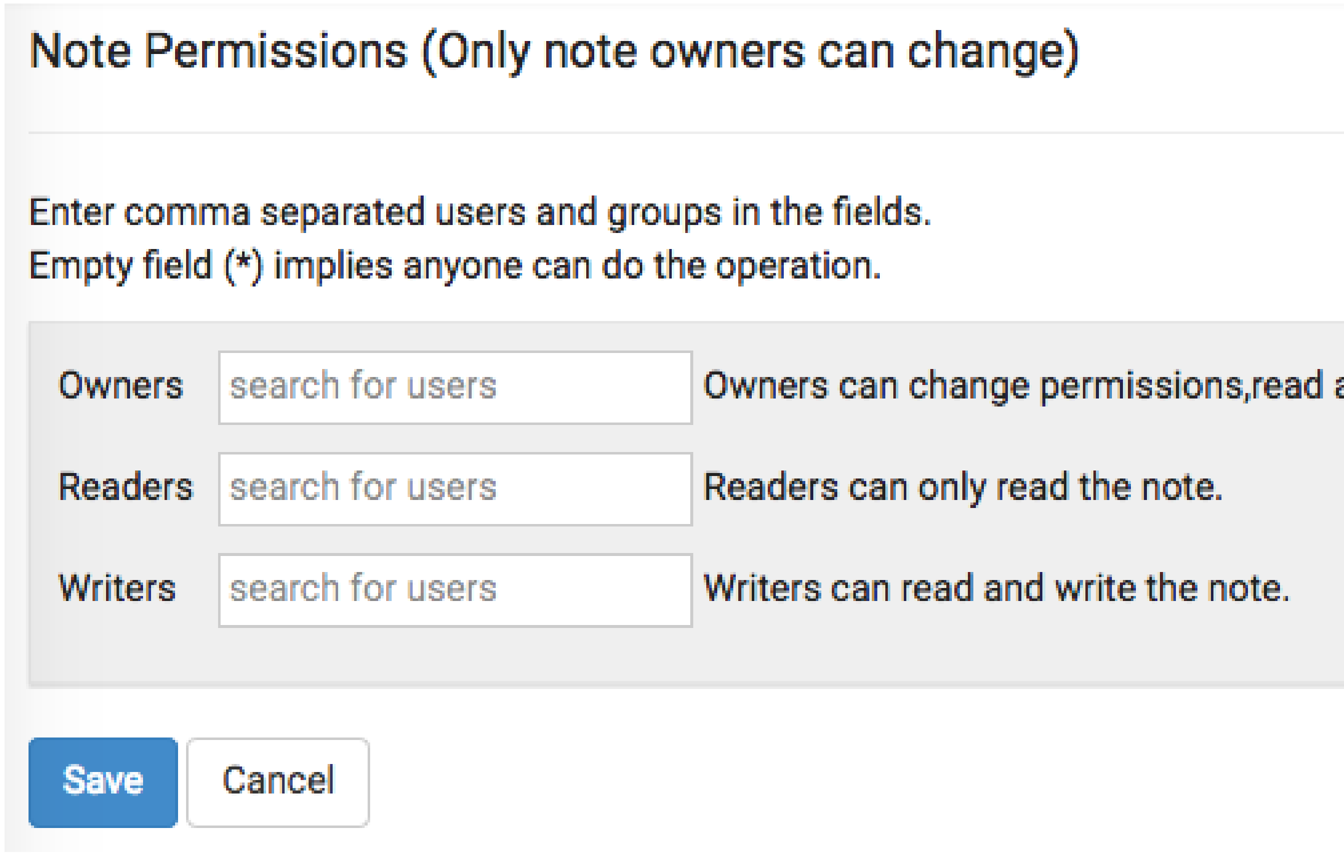
2. The owner of the notebook should navigate to the note and complete the following steps:

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 a
Readers	search for users	Readers can only read the note.
Writers	search for users	Writers can read and write the note.

Enable Access Control for Data

This section describes how to restrict access to Apache Zeppelin 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.

Shiro Settings: Reference

This section provides additional information about the Shiro settings used to configure Apache Zeppelin security.

Active Directory Settings

This section provides additional information about Shiro 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.

activeDirectoryRealm

specifies the class name to use for AD authentication. You should set this to org.apache.zeppelin.realm.ActiveDirectoryGroupRealm.

activeDirectoryRealm.url

specifies the host and port where Active Directory is set up. .

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

defines 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.

LDAP Settings

This section provides additional information about Shiro 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.

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. ..

ldapRealm.contextFactory.environment[ldap.searchBase]

defines the base distinguished name from which the LDAP search starts. Shiro searches for user DnTemplate 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

define 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=h
adoop,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.

General Settings

This section provides additional information about Shiro 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.

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.

Before you begin

In this example, the corresponding account information is configured in Active Directory (at adhost.field.hortonworks .com) 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
activeDirectoryRealm.systemUsername
activeDirectoryRealm.systemPassword

# 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
```

Using Apache Zeppelin

How to configure Apache Zeppelin.

Introduction

An Apache Zeppelin notebook is a browser-based GUI you can use for interactive data exploration, modeling, and visualization.

Important: Zeppelin will be deprecated in Cloudera Runtime 7.2.18. For more information, see *Deprecation Notices in Cloudera Runtime 7.2.18*.

As an Apache Zeppelin 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.

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 built-in 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.

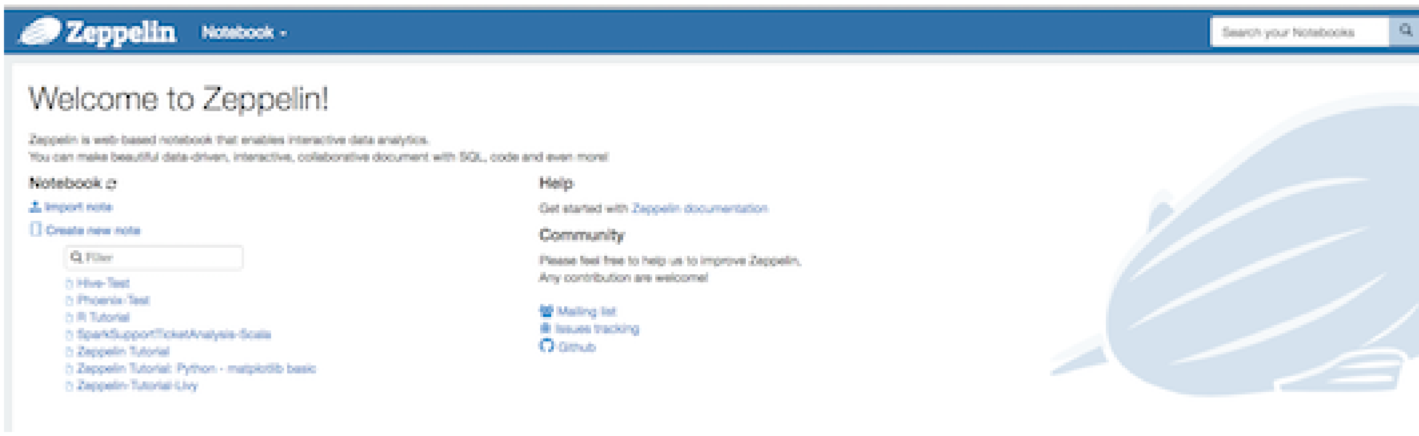
Launch Zeppelin

Use the following steps to launch Apache Zeppelin.

To launch Zeppelin in your browser:

1. In the Cloudera Data Platform (CDP) Management Console, go to Data Hub Clusters.
2. Find and select the cluster you want to use.
3. Click the Zeppelin link.

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!

Notebook

Import note
Create new note
Filter
Hive-Test
Phoenix-Test
R Tutorial
SparkSupportTicketAnalysis-Scala
Zeppelin Tutorial
Zeppelin Tutorial: Python - matplotlib basic
Zeppelin-Tutorial-Livy

Help

Get started with Zeppelin documentation

Community

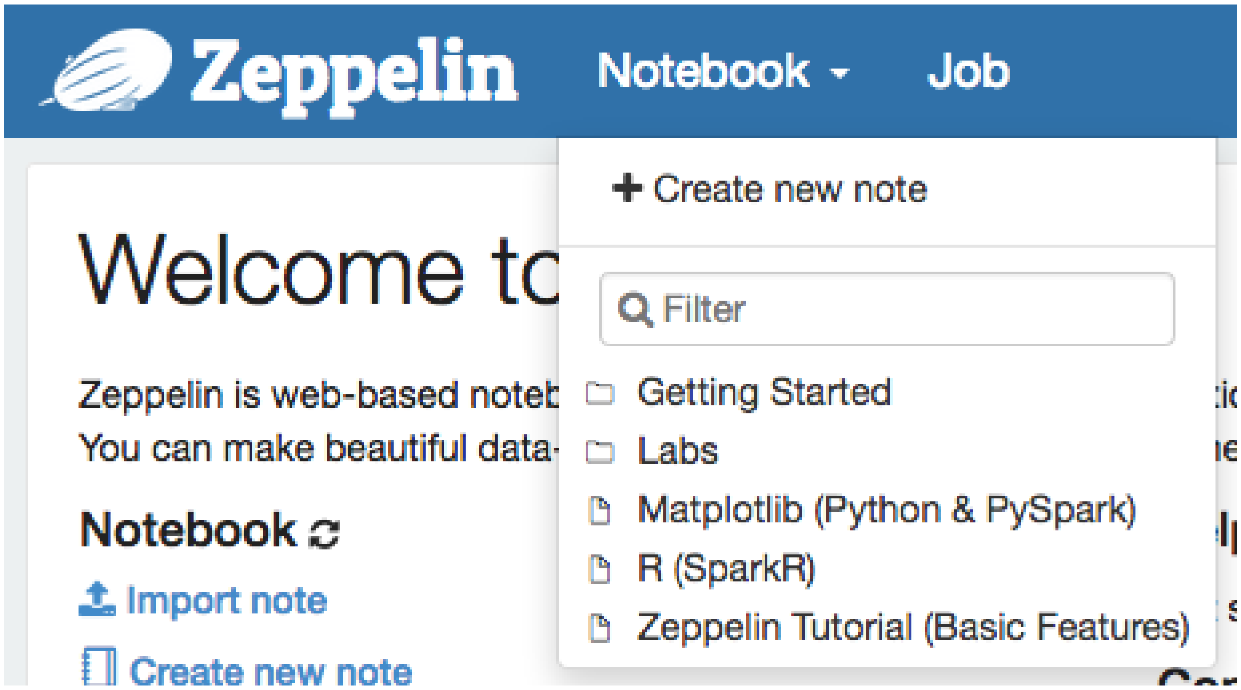
Please feel free to help us to improve Zeppelin.
Any contribution are welcome!

Mailing list
Issues tracking
Github

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

- Notebook

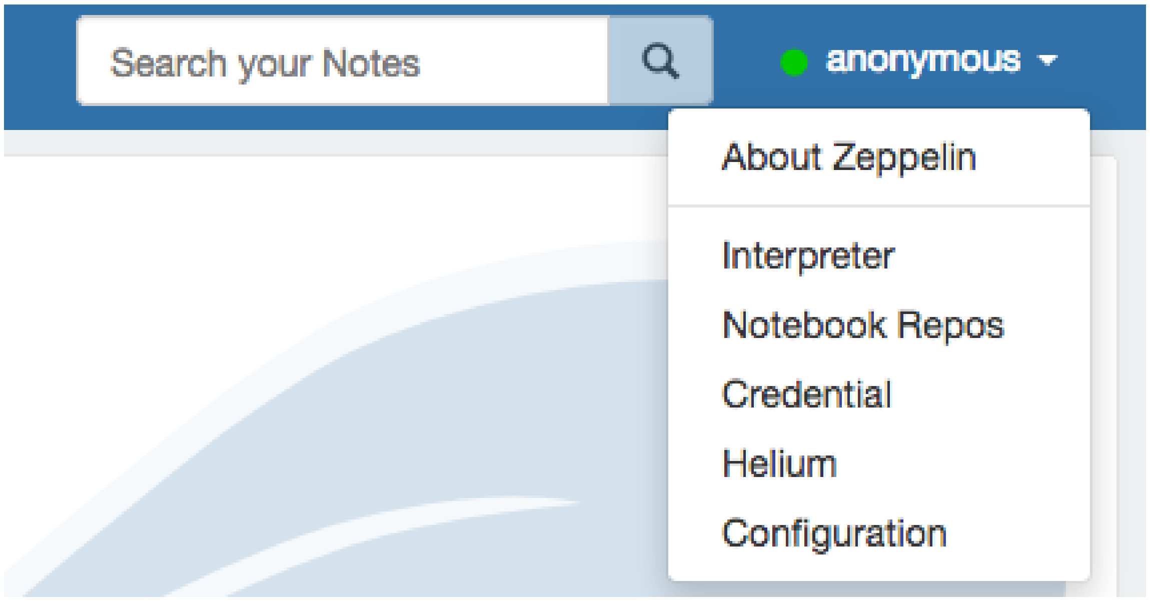
  Open a note, filter the list of notes by name, or create a note:



Zeppelin
Notebook
Job
Create new note
Filter
Getting Started
Labs
Matplotlib (Python & PySpark)
R (SparkR)
Zeppelin Tutorial (Basic Features)
Welcome to
Zeppelin is web-based noteb
You can make beautiful data-
Notebook
Import note
Create new note

- User settings

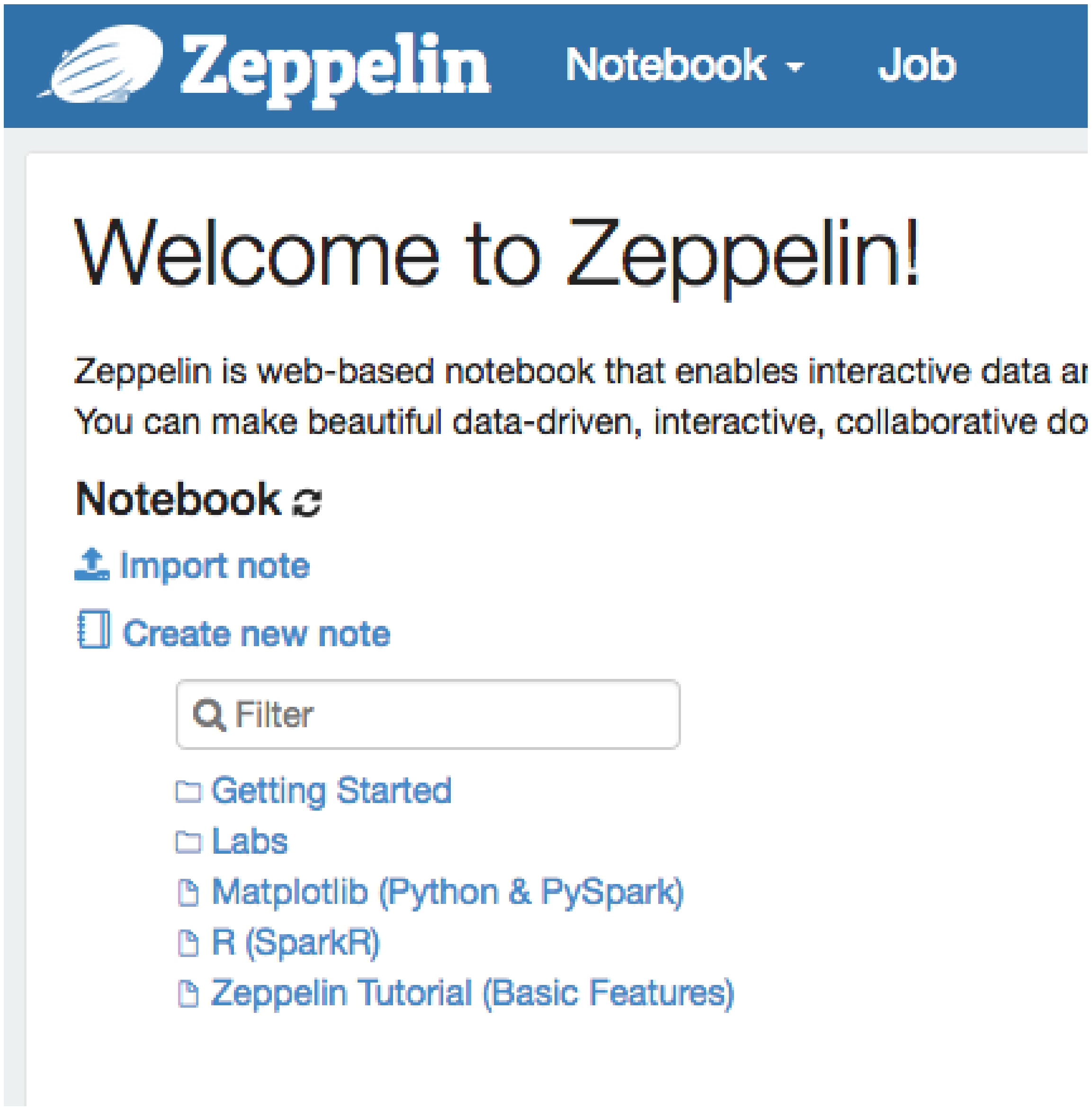
  Displays your username, or "anonymous" if security is not configured for Zeppelin.



  - 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!

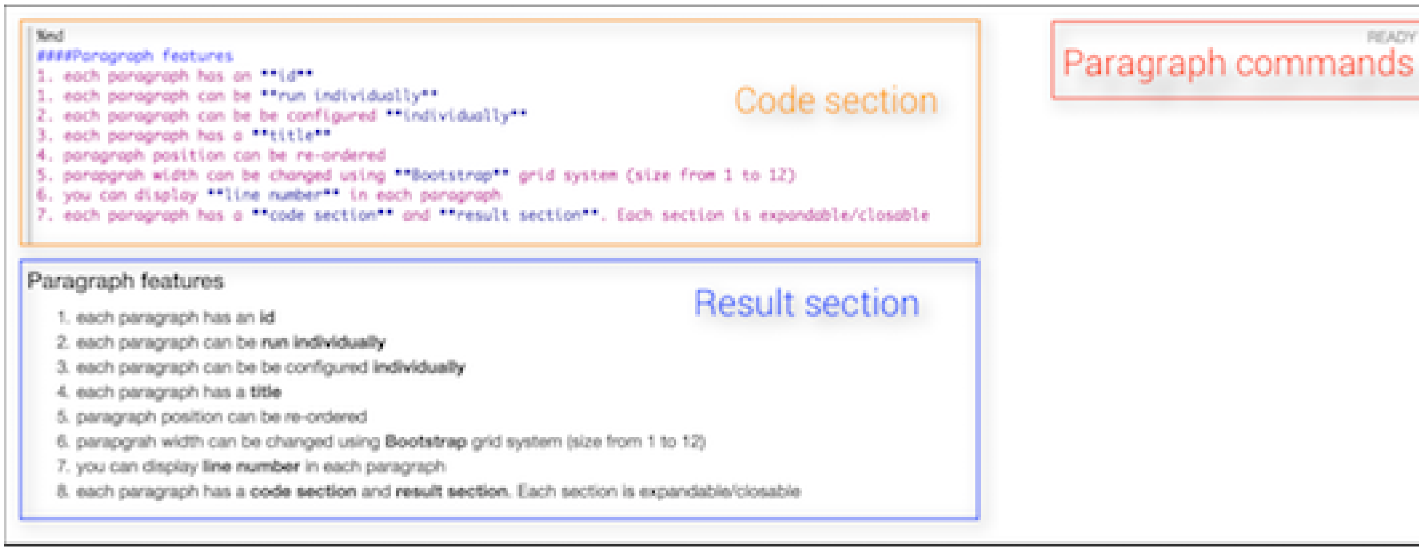
Zeppelin is web-based notebook that enables interactive data ar
You can make beautiful data-driven, interactive, collaborative do

Working with Zeppelin Notes

This section provides an introduction to Apache Zeppelin notes.

An Apache Zeppelin 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.

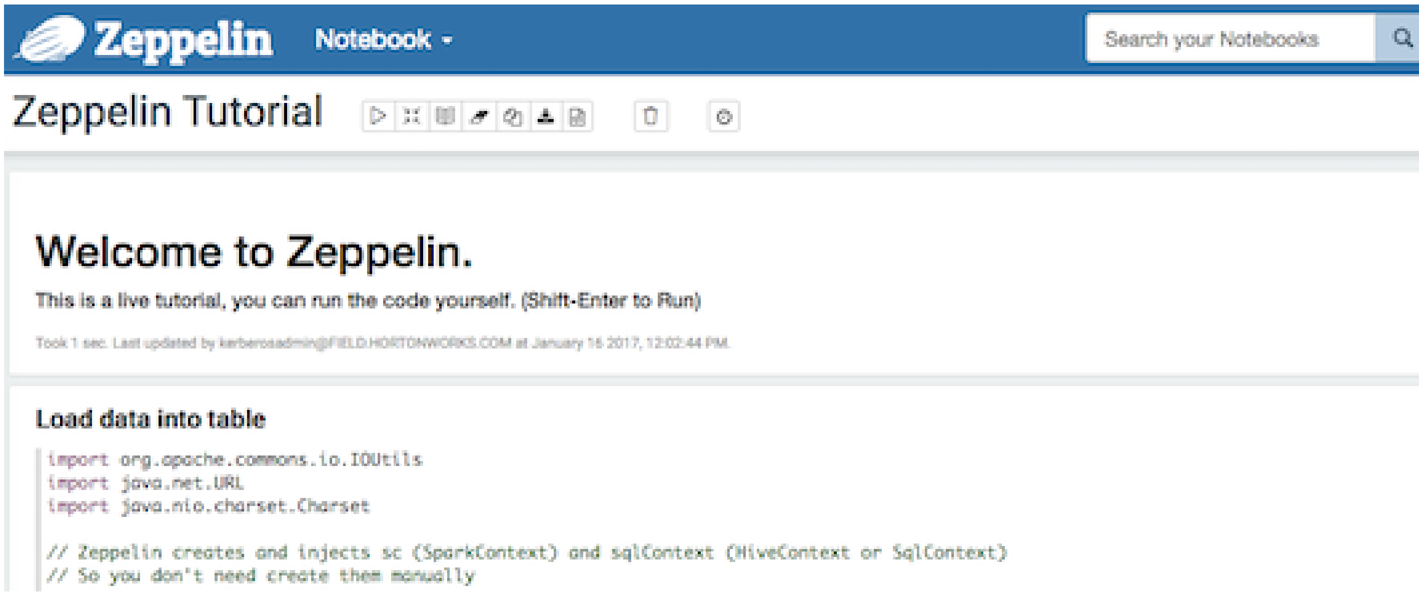


```
%md
####Paragraph features
1. each paragraph has an **id**
1. each paragraph can be **run individually**
2. each paragraph can be be configured **individually**
3. each paragraph has a **title**
4. paragraph position can be re-ordered
5. parapgrah width can be changed using **Bootstrap** grid system (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.

Create and Run a Note

Use the following steps to create and run an Apache Zeppelin 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.

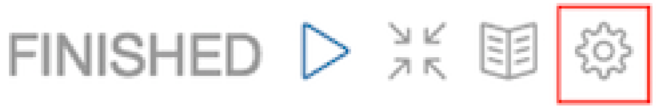
When you create a note, it appears in the list of notes on the left side of the home page and in the Notebook menu. By default, Zeppelin stores notes in the $ZEPPELIN_HOME/notebook folder.

To run your code:

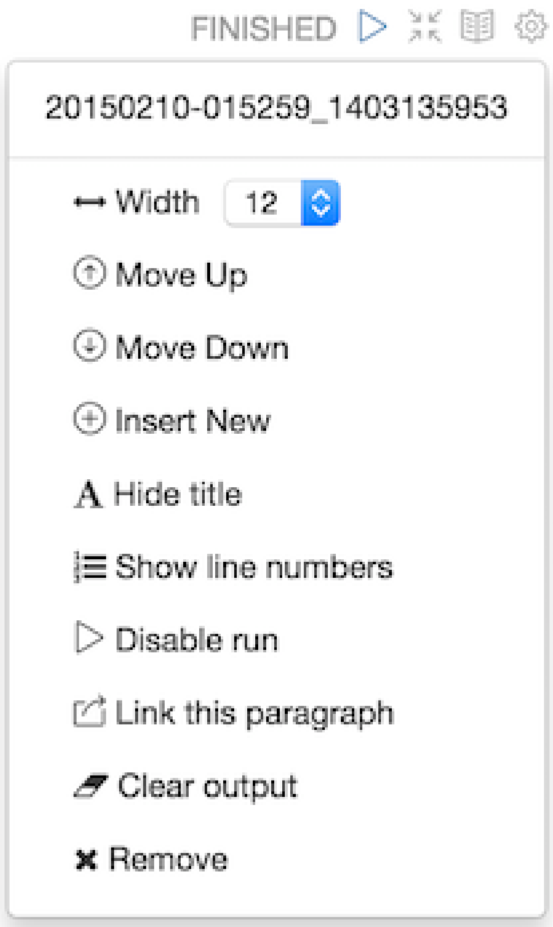
1. Click the triangle button in the cell that contains your code:

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.



Import a Note

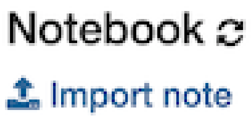
Use the following steps to import an Apache Zeppelin note.

About this task

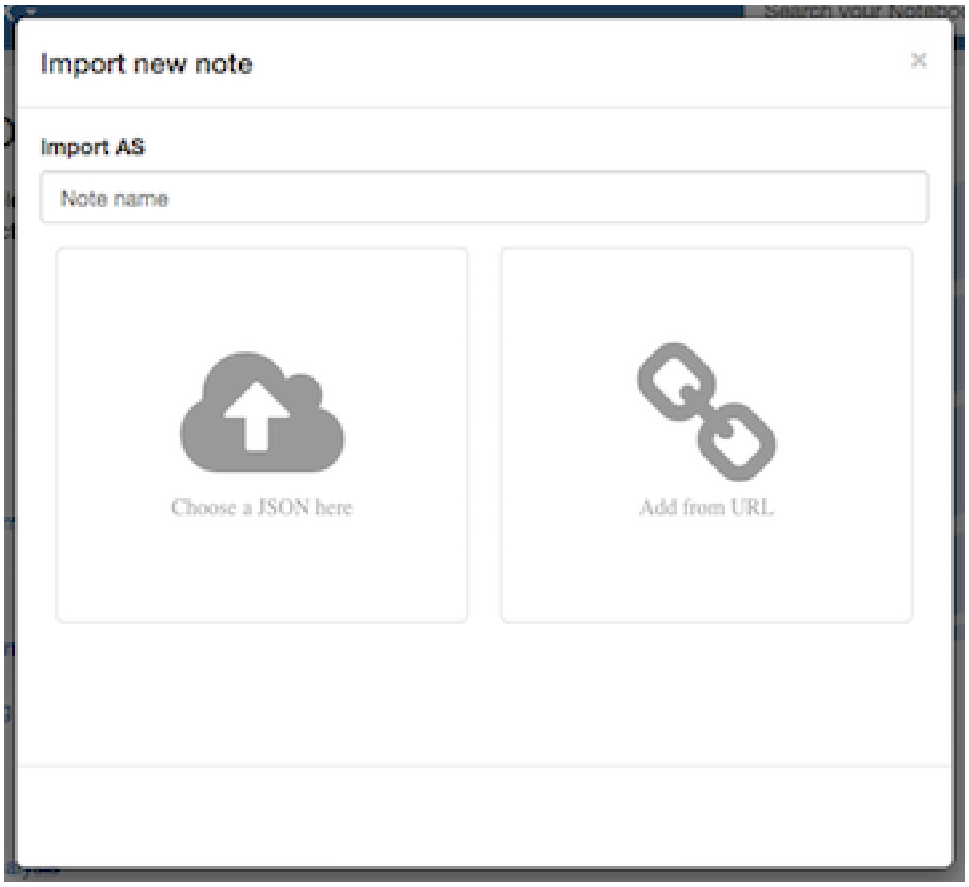
To import a note from a URL or from a JSON file in your local file system:

Procedure

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



2. Zeppelin displays an import dialog box:



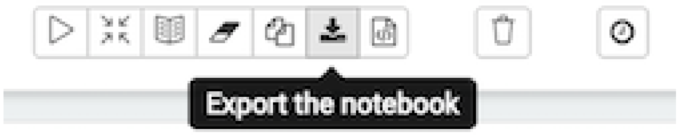
3. To upload the file or specify the URL, click the associated box.

   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.

Export a Note

Use the following steps to export an Apache Zeppelin 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.

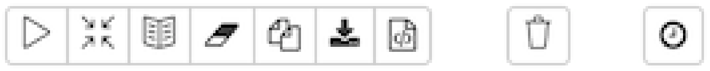
Note: Zeppelin exports code and results sections in all paragraphs. If you have a lot of data in your results sections, consider trimming results before exporting them.

Using the Note Toolbar

This section describes how to use the Apache Zeppelin 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:

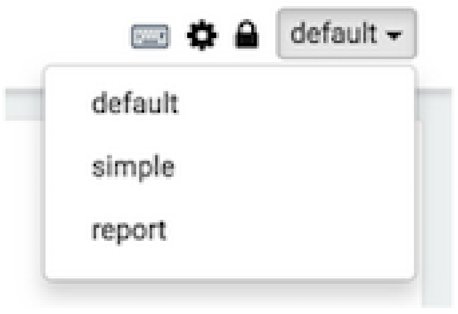


These buttons perform the following operations:

- Run 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.

Import External Packages

Use the following steps to import external packages into Apache Zeppelin.

To use an external package within a Zeppelin note, you can use one of the following approaches:

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

  For more information, see the Dependency Management for Interpreter documentation 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.dat
    abricks:spark-csv_2.10:1.2.0 spark.files /path/mylib1.py,/path/mylib2.eg
    g,/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" in the HDP Apache Spark guide.

Configuring and Using Zeppelin Interpreters

An Apache Zeppelin 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 Cloudera Data Platform supports the following interpreters:

- JDBC (supports Hive).

  However, do note that the JDBC interpreter for Apache Hive is deprecated.
- Markdown
- Livy (supports Spark, Spark SQL, PySpark, PySpark3, and SparkR)

  Note: If you are upgrading from HDP to , ensure that your existing notebooks are manualy migrated to the Livy interpreter. Support for the Spark interpreter is no longer available.
- AngularJS

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

Modify interpreter settings

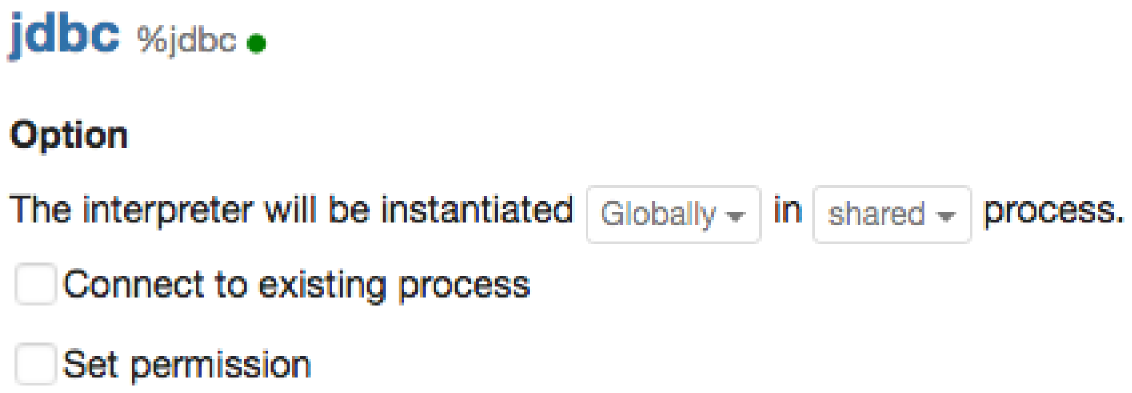
Use the following steps to modify Apache Zeppelin interpreter settings.

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.

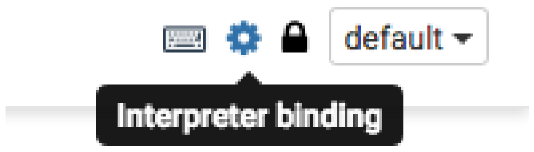
Note: The Interpreter page is subject to access control settings. If the page does not list a set of interpreters, check with your system administrator.

Using Zeppelin Interpreters

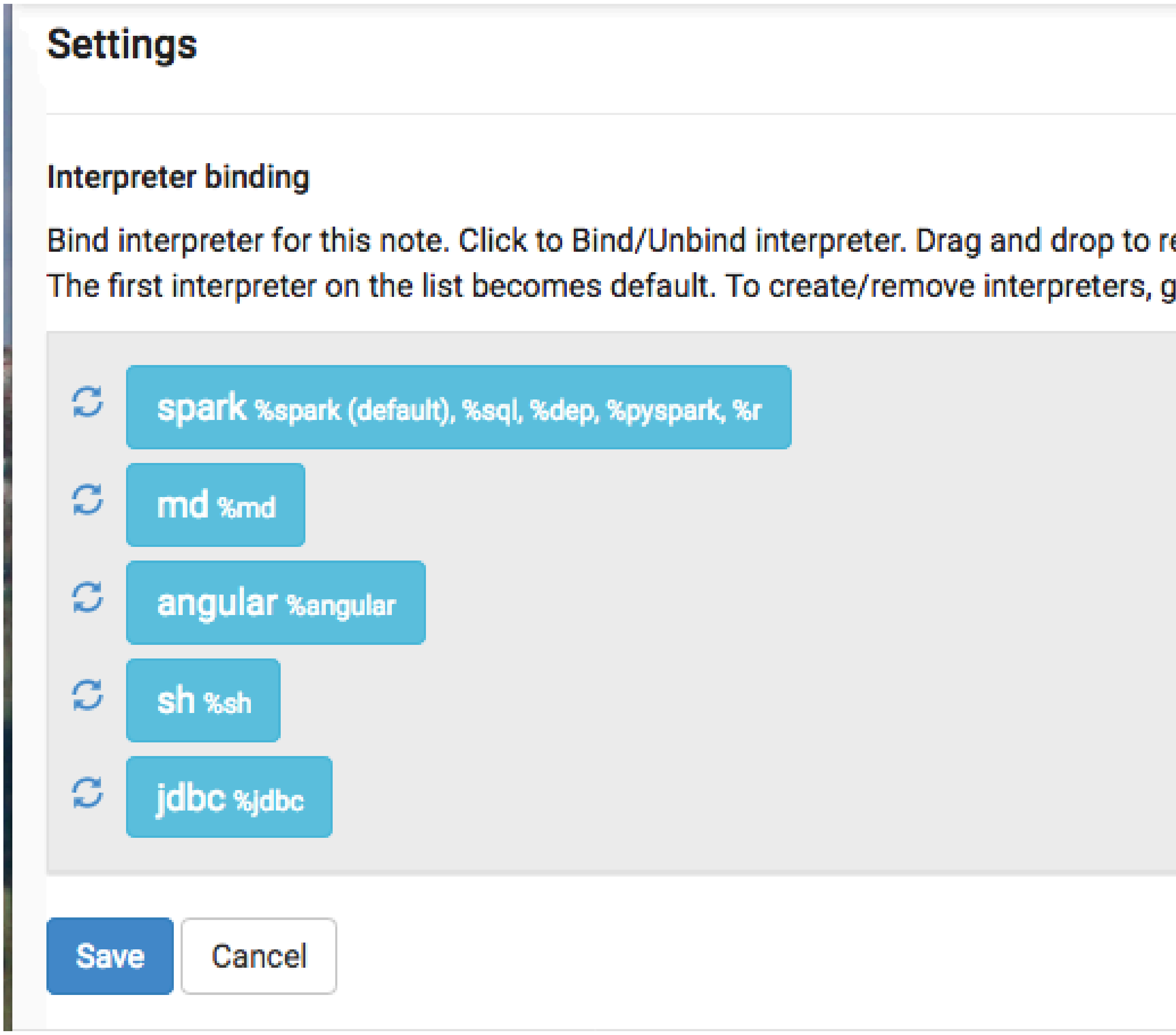
This section describes how to use Apache Zeppelin interpreters.

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

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



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



Settings

Interpreter binding

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

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".

Use an interpreter in a paragraph

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

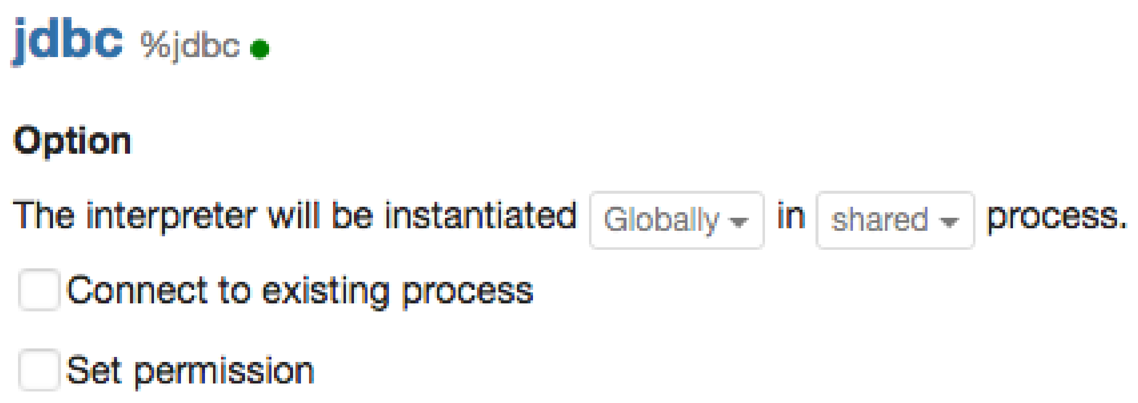
The following paragraph uses the %sh interpreter to access the system shell and list the current working directory:

```
%sh
```

```
pwd
home/zeppelin
```

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

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


jdbc %jdbc

Option

The interpreter will be instantiated Globally in shared process.

Connect to existing process

Set permission

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

Use interpreter groups

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

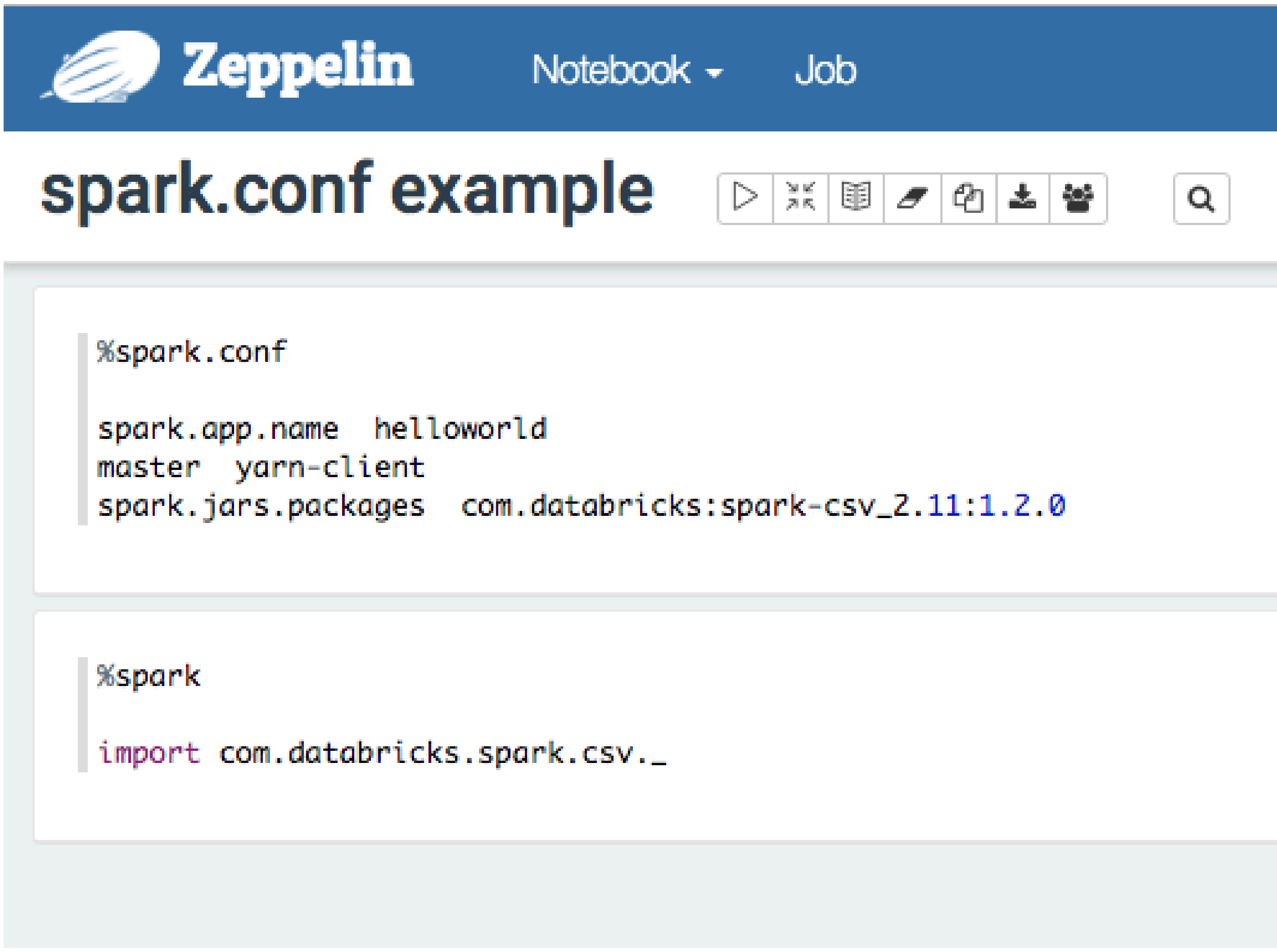
Customize interpreter settings in a note

This section describes how to customize Apache Zeppelin interpreter settings on a per-note basis.

About this task

You can use the Zeppelin conf interpreter to customize interpreter configuration settings on a per-note basis. The conf interpreter is a generic interpreter that can be used to customize any Zeppelin interpreter on a per-note basis.

In the following example, zeppelin_custom_note_conf.png to customize the Spark interpreter in a Note.



```
%spark.conf

spark.app.name  helloworld
master  yarn-client
spark.jars.packages  com.databricks:spark-csv_2.11:1.2.0
```

```
%spark

import com.databricks.spark.csv._
```

First paragraph:

```
%spark.conf
spark.app.namehelloworld
master  yarn-client
spark.jars.packages com.databricks:spark-csv_2.11:1.2.0
```

Second paragraph:

```
%spark

import com.databricks.spark.csv._
```

In the first paragraph, the conf interpreter is used to create a custom Spark interpreter configuration (set app name, yarn-client mode, and add spark-csv dependencies). After running the first paragraph, the second paragraph can be run to use spark-csv in the note.

In order for the conf interpreter to run successfully, it must be configured on an isolated per-note basis. Also, the paragraph with the conf interpreter customization settings must be run first, before subsequent applicable interpreter processes are launched.

Use the JDBC interpreter to access Hive

This section describes how to use the Apache Zeppelin JDBC interpreter to access Apache 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" in this guide.

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 /opt/cloudera/parcels/<version>/lib/zeppelin/interpreter/<name_of_interpreter> (or create a soft link).

   For example, /opt/cloudera/parcels/<version>/lib/zeppelin/interpreter/livy/
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 tohive.server2.proxy.user.
4. Click Save, then click restart to restart the JDBC interpreter.

Use the Livy interpreter to access Spark

This section describes how to use the Livy interpreter to access Apache 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 HDP Apache Spark guide.
- Before using Livy in a note, check the Interpreter page to ensure that the Livy interpreter is configured properly for your cluster.

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, see "Submitting Spark Applications Through Livy" in the HDP Apache Spark guide.

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 interepreter in yarn-cluster mode, 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

Using Spark Hive Warehouse and HBase Connector Client .jar files with Livy

This section describes how to use Spark Hive Warehouse Connector (HWC) and HBase-Spark connector client .jar files with Livy. These steps are required to ensure token acquisition and avoid authentication errors.

Use the following steps to use Spark HWC and HBase-Spark client .jar files with Livy:

1. Copy the applicable HWC or HBase-Spark .jar files to the Livy server node and add these folders to the livy.file.local-dir-whitelist property in the livy.conf file.
2. Add the required Hive and HBase configurations in the Spark client configuration folder:

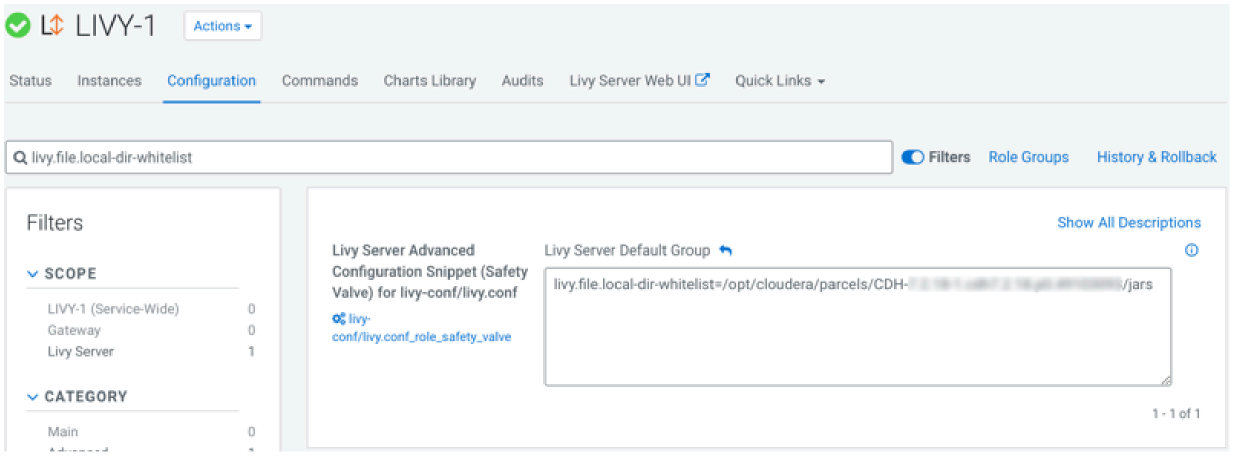
   - Hive: /etc/spark3/conf/hive-site.xml
   - HBase: /etc/spark3/conf/hbase-site.xml

   Or add the required configurations using the conf field in the session creation request. This is equivalent to using "--conf" in spark-submit.
3. Reference these local .jar files in the session creation request using the file:/// URI format.

HWC Example

1. In Cloudera Manager, go to Clusters Livy .
2. Click the Configuration tab and search for the "Livy Server Advanced Configuration Snippet (Safety Valve) for livy-conf/livy.conf" property.

3. Add the /opt/cloudera/parcels/CDH-*[*** CLOUDERA VERSION ***]*/jars folder to the livy.file.local-dir-whitelist property in the livy.conf file.



livy.file.local-dir-whitelist

Livy Server Advanced Configuration Snippet (Safety Valve) for livy-conf/livy.conf
livy-conf/livy.conf_role_safety_valve

Livy Server Default Group

```
livy.file.local-dir-whitelist=/opt/cloudera/parcels/CDH-[redacted]/jars
```

4. Save the changes and restart the Livy service.
5. Log in to the Zeppelin Server Web UI and restart the Livy interpreter.
6. When running the Zeppelin Livy interpreter, reference the HWC .jar file as shown below.

   - For Spark 3:

```
%livy2.conf
livy.spark.jars file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION
 ***]/jars/hive-warehouse-connector-spark3-assembly-[*** PARCEL VERSION
 ***].jar
```

HBase-Spark Connector Example

1. When running the Zeppelin Livy interpreter, reference the following HBase .jar files as shown below. Note that some of these .jar files have 644/root permissions, and therefore may throw an exception. If this happens, you may need to change the permissions of the applicable .jar files on the Livy node.

```
%livy.conf
livy.spark.jars file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION
 ***]/jars/hbase-spark3-[*** PARCEL VERSION ***].jar
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
shaded-protobuf-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
shaded-miscellaneous-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
protocol-shaded-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
shaded-netty-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
shaded-client-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
shaded-mapreduce-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
common-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
server-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
client-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
protocol-[*** PARCEL VERSION ***].jar,
file:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/hbase-
mapreduce-[*** PARCEL VERSION ***].jar,
ile:///opt/cloudera/parcels/CDH-[*** CLOUDERA VERSION ***]/jars/guava-32.1
.3-jre.jar
```

Note: The references to HBase-Spark connector jar file are included because HBase-Spark connector was used in this example. They are not required for token acquisition.