Apache Zeppelin 3

Configuring Apache Zeppelin Security

Date of Publish: 2018-04-01

http://docs.hortonworks.com

Contents

Introduction

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

About this task

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

Procedure

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.

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.

Getting Started

Use the following steps to begin configuring Apache Zeppelin security.

Before you begin

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.

Configure 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.

Configure Security on a non-Ambari Cluster

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 the shiro.ini file as described in the following subsections.

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

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

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.

Related Information

Apache Shiro Configuration

Configure Zeppelin for Authentication: Non-Production Use

This section describes how to configure Apache Zeppelin quickly for non-production use.

About this task

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:

Procedure

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

Configure Zeppelin for Authentication: LDAP and Active Directory

This section describes how to configure Apache Zeppelin for LDAP and AD authentication.

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.

Configuring Authentication for Production Using Active Directory

Use the following steps to configure Apache Zeppelin for Active Directory in production environments.

About this task

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

Procedure

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 the shiro.ini file, 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. Setzeppelin.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:

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

What to do next

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
```

Related Information

Shiro Authentication for Apache Zeppelin

Configuring Authentication for Production Using LDAP

Use the following steps to configure Apache Zeppelin for LDAP in production environments.

About this task

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.

Procedure

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
   ```

Related Information

Shiro Authentication for Apache Zeppelin

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 [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=n
# 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 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

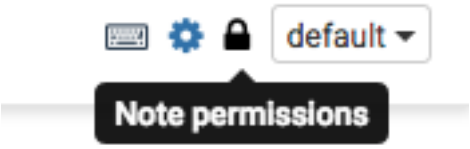
There are two main steps in this process: defining the searchBase property in the Zeppelin Shiro configuration, and then specifying permissions.

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 searchBase controls where Zeppelin looks for users and groups.

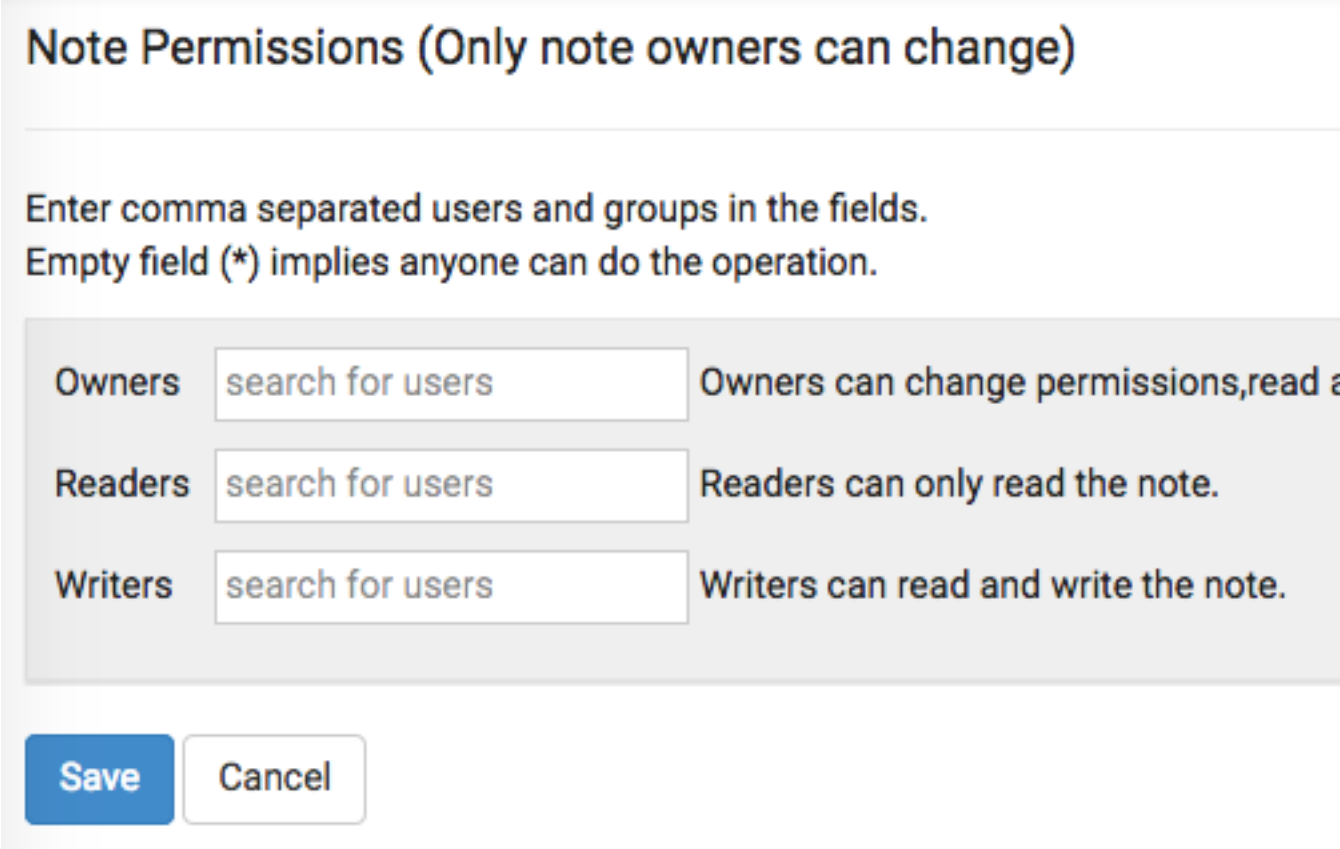
   For more information, refer to "Shiro Settings: Reference" in this guide. For an example, see "Configure Zeppelin for Authentication: LDAP and Active Directory" in this guide.
2. The owner of the notebook should navigate to the note and complete the following steps:

   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.



Related Information

Apache Zeppelin Notebook Authorization

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.

Configure SSL for Zeppelin

Use the following steps to configure SSL for Apache Zeppelin.

Procedure

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:

   If you wish to use client-side certificate authentication, enable client-side authentication and configure the associated trust store settings:
5. Check to make sure that all settings are valid.
6. Connect using HTTPS.

What to do next

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

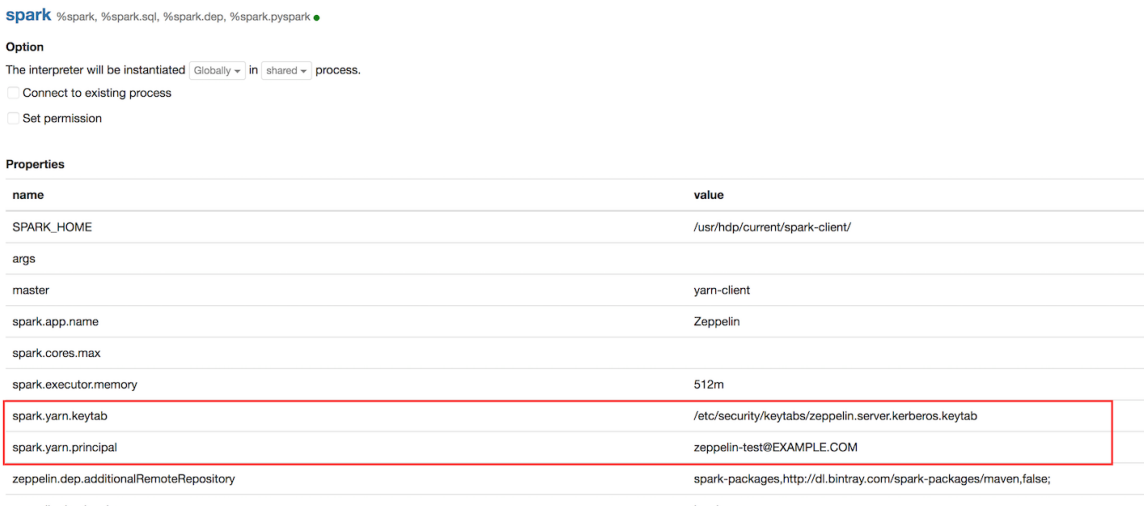
Configure Zeppelin for a Kerberos-Enabled Cluster

Use the following steps to configure Apache Zepellin 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" in the HDP Apache Ambari Security guide.
- 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 HDP 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.



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

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.

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 HDP Command Line Upgrade guide.

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.

Related Information
Apache Shiro Realms

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 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

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=gues
```

```
ldapRealm.contextFactory.systemPassword=somePass
```

ldapRealm.authorizationCachingEnabled

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

Related Information
LDAP Realm Settings
Apache Shiro Realms

General Settings

This section provides additional information about Shiro generas 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

# 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
```