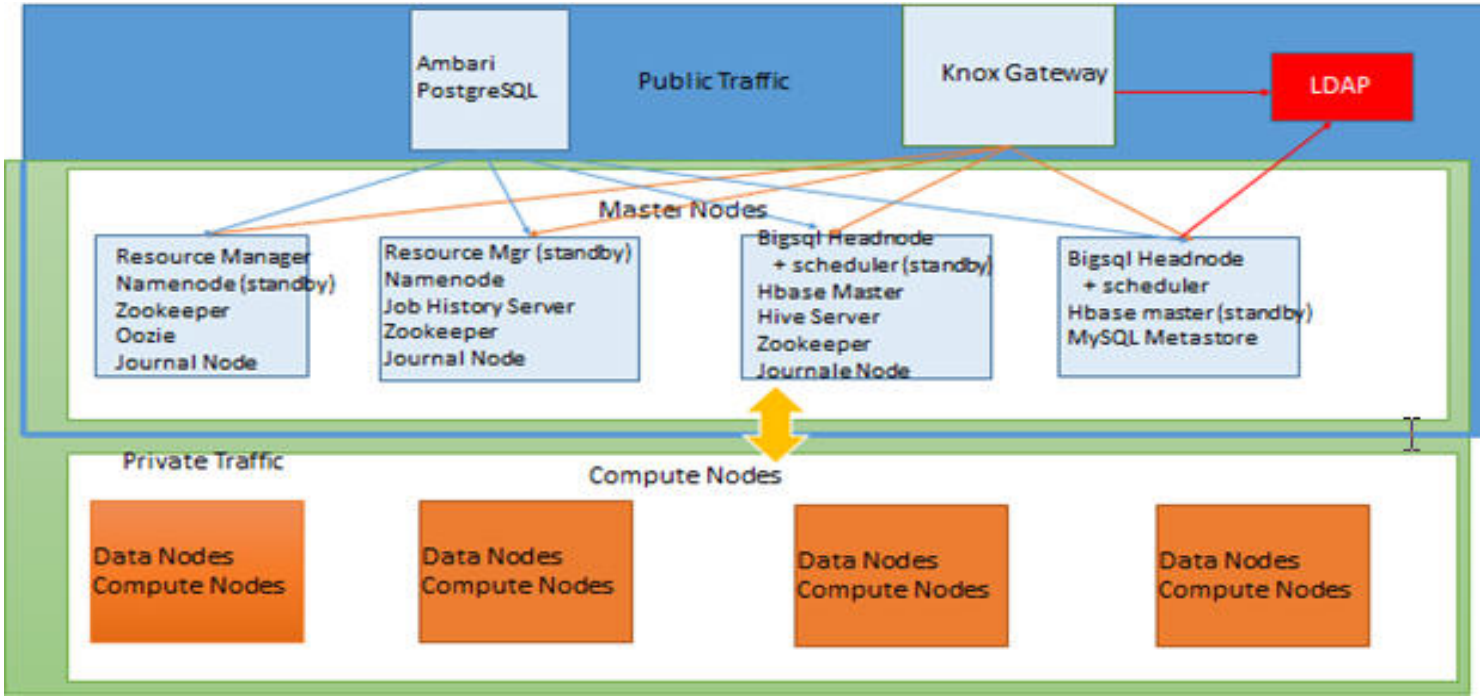
Apache Knox

Apache Knox Gateway Overview

The Apache Knox Gateway (“Knox”) is a system to extend the reach of Apache™ Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop Security.
Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. The Knox Gateway is designed as a reverse proxy.

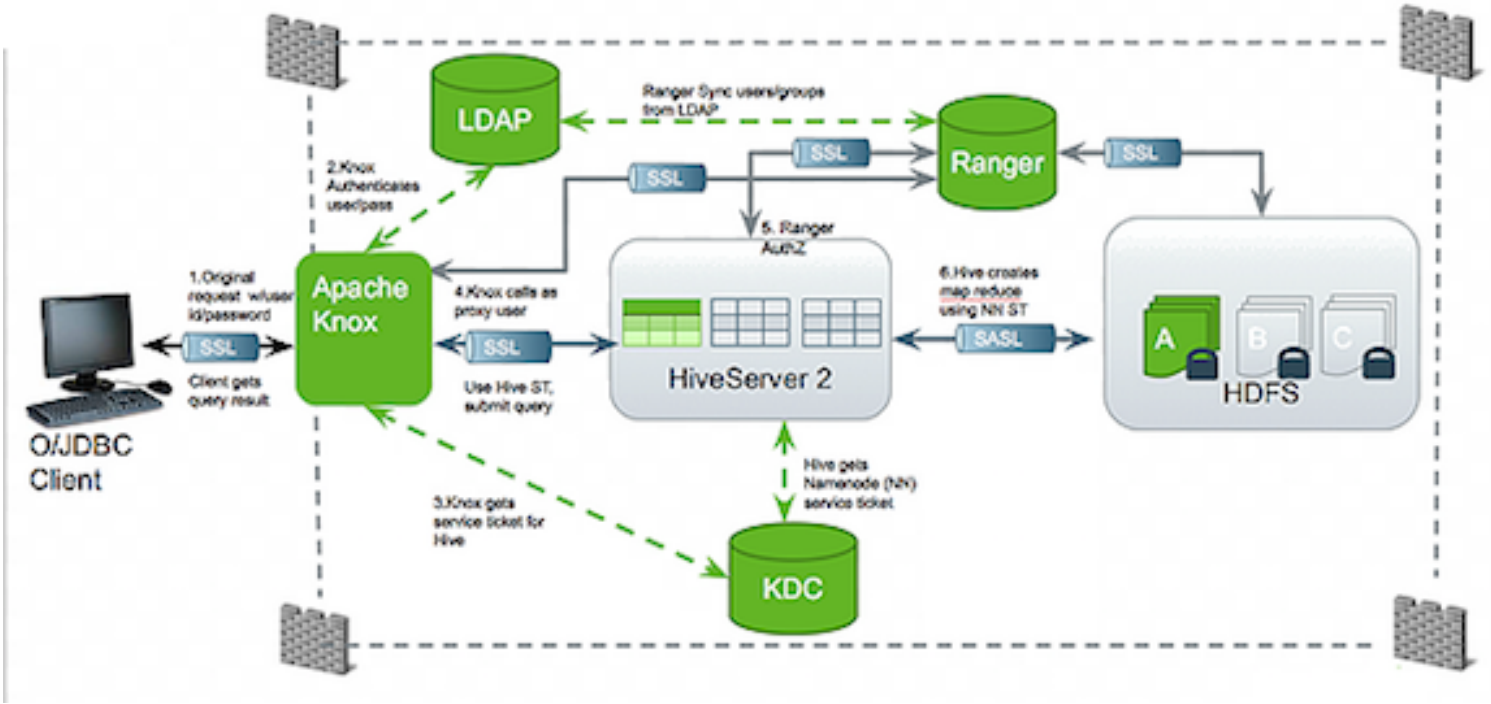
The key feature of Knox Gateway is that it provides perimeter security for Hadoop REST APIs by limiting the network endpoints required to access a Hadoop cluster. Thus, it hides the internal Hadoop cluster topology from end users.

Knox integrates with Identity Management and SSO systems used in enterprises and allows identity from these systems be used for access to Hadoop clusters



Knox Gateways provides security for multiple Hadoop clusters, with these advantages:

- **Simplifies access**: Extends Hadoop's REST/HTTP services by encapsulating Kerberos to within the Cluster.
- **Enhances security**: Exposes Hadoop's REST/HTTP services without revealing network details, providing SSL out of the box.
- **Centralized control**: Enforces REST API security centrally, routing requests to multiple Hadoop clusters.
- **Enterprise integration**: Supports LDAP, Active Directory, SSO, SAML and other authentication systems.



Knox can be used with both unsecured Hadoop clusters, and Kerberos secured clusters. In an enterprise solution that employs Kerberos secured clusters, the Apache Knox Gateway provides an enterprise security solution that:

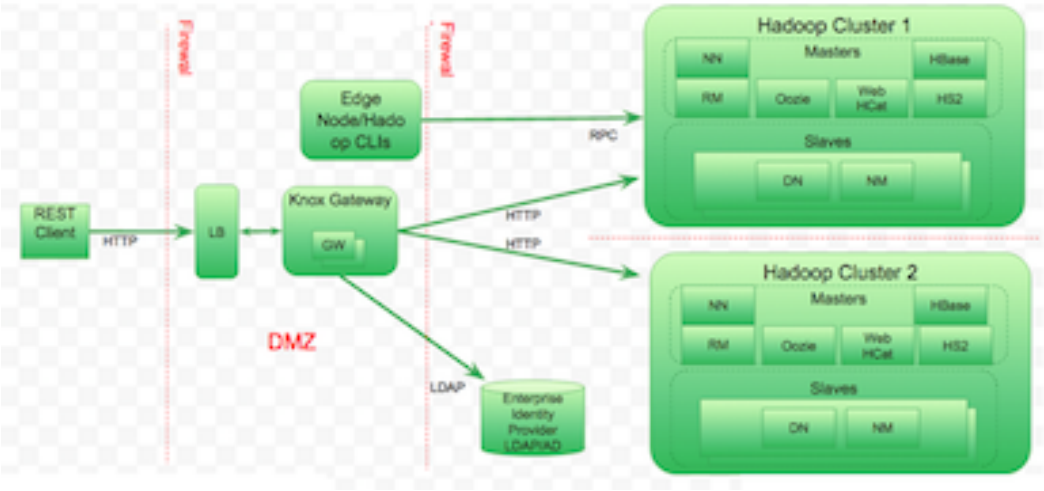
- Integrates well with enterprise identity management solutions
- Protects the details of the Hadoop cluster deployment (hosts and ports are hidden from end users)
- Simplifies the number of services with which a client needs to interact

In short, the knox works as a proxy by hiding the actual structure of the Hadoop cluster and providing a single point of access to all the services. For example, check the access methods to access webHDFS to get status of the directory /user using Knox gateway and without using Knox gateway.

Knox Gateway Deployment Architecture

Users who access Hadoop externally do so either through **Knox**, via the Apache **REST API**, or through the **Hadoop CLI** tools.

The following diagram shows how Apache Knox fits into a Hadoop deployment.



Supported Hadoop Services

Apache Knox Gateway supports the following Hadoop services versions in both Kerberized and Non-Kerberized clusters:

Supported Component APIs: Proxy	Supported Component UIs: Proxy
• YARN • WebHDFS • WebHCat/Templeton • Oozie • HBase/Stargate • Hive (via WebHCat) • Hive (via JDBC) • Ambari • Atlas • Ranger • Zeppelin	• Ambari UI • Atlas • Ranger Admin Console • Zeppelin

Please refer Knox official website and Hortonworks official documents to know more about the currect supported services.
https://knox.apache.org/
https://docs.hortonworks.com/

Install Knox (Through Ambari)

Ambari Dashboard > Add services > Knox > Select node to deploye Knox > Choose Master Secret Key > Deploy

Master secret key will be encrypted & stored in a file `/var/lib/knox/data-*/security/master`
The master secret is required to start the gateway. The secret protects artifacts used by the gateway instance, such as the keystore, trust stores and credential stores.

Defining Cluster Topologies

- The Knox Gateway supports one or more Hadoop clusters.
- Each Hadoop cluster configuration is defined in a topology deployment descriptor file which can be found in `/etc/knox/conf/topologies` directory and is deployed to a corresponding WAR file modified timestamp in `/var/lib/knox/data-*/deployments` directory.
- These files define how the gateway communicates with each Hadoop cluster.
- You can configure topology through Ambari *(Ambari > Knox > Config > Advance Topology)*

Cluster topology descriptors have the following XML format:

```
<topology>
    <gateway>
        <provider>
            <role></role>
            <name></name>
            <enabled></enabled>
            <param>
                <name></name>
                <value></value>
            </param>
        </provider>
    </gateway>
    <service></service>
</topology>
```

To know more about each Provider and Service Role, Kindly refer below Given Link
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/defining_cluster_topologies.html

Configuring a Hadoop Server for Knox

The Apache Knox Gateway redirects external requests to an internal Hadoop service using service name and URL of the service definition.

Setting up Hadoop Service URLs

To configure access to an internal Hadoop service through the Knox Gateway:
Add an entry in cluster topology file similar to the following, for each Hadoop service:

```
<topology>
     <service>
         <role>$service_name</role>
         <url>$schema://$hostname:$port</url>
     </service>
</topology>
```

where:

- $service_name is: **AMBARI**, **AMBARIUI**, **ATLAS**, **HIVE**, **JOBTRACKER**, **NAMENODE**, **OOZIE**, **RANGER**, **RANGERUI**, **RESOURCEMANAGER**, **WEBHBASE**, **WEBHCAT**, **WEBHDFS**, **DRUID-COORDINATOR, DRUID-OVERLORD-UI, DRUID-OVERLORD, DRUID-ROUTER, DRUID-BROKER**, **ZEPPELINUI**, or **ZEPPELINWS**
- <url> is the complete internal cluster URL required to access the service, including:
  - $schema -- the service protocol
  - $hostname -- the resolvable internal host name
  - $port -- the service listening port

Validating Service Connectivity (DEMO-LDAP)

1_ Start demo ldap from Ambari Dashboard
Ambari > Knox > Service Action > Start DemoLDAP

2_ Access WebHDFS using *HDFS API(**without Knox Gateway**)* and ***Using Knox Gateway***

a) Using WebHDFS API

```
# curl -k "http://namenode.hadoop.com:50070/webhdfs/v1/user?op=LISTSTATUS"
```

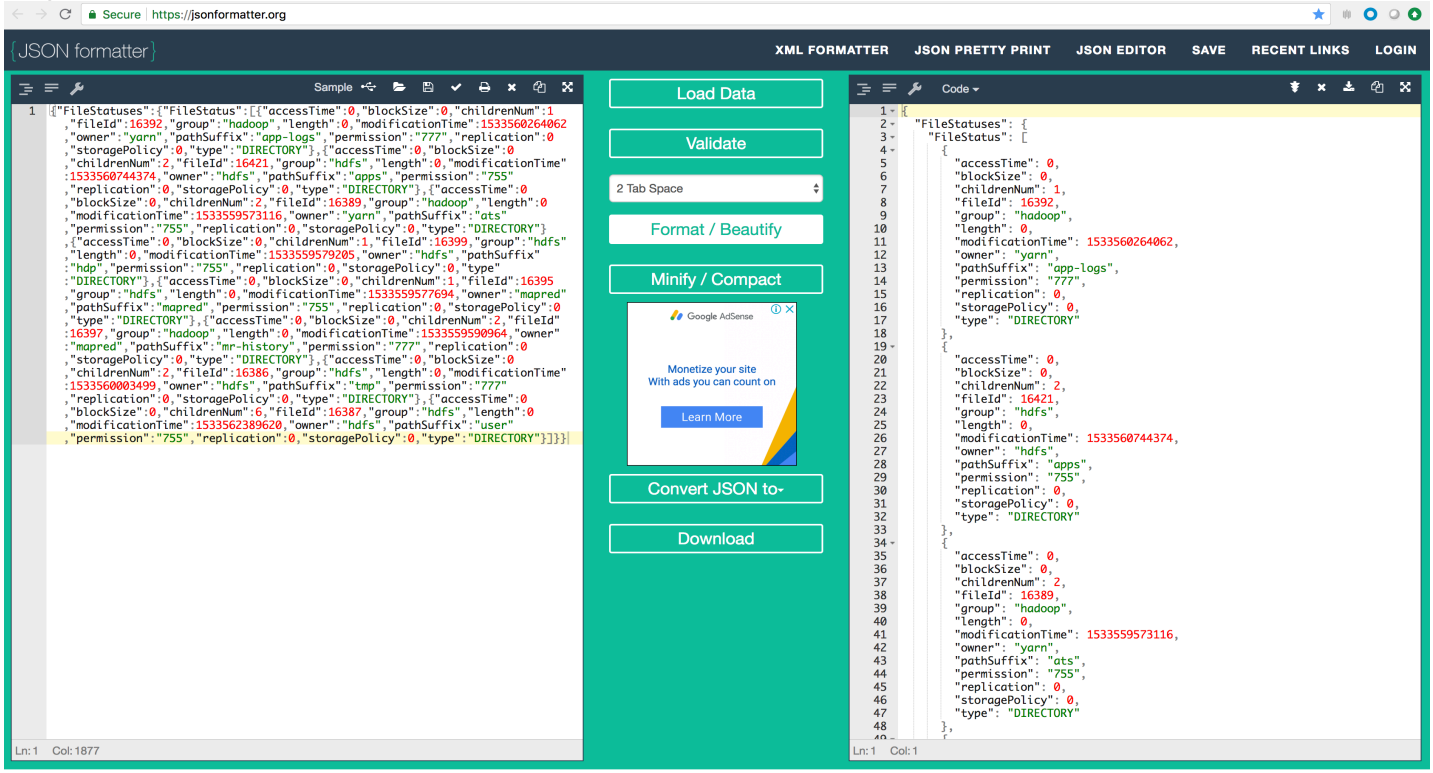
b) Using Knox gateway

```
# curl -k -u guest:guest-password "https://knox.hadoop.com:8443/gateway/default/webhdfs/v1/user?op=LISTSTATUS"
```

IMPORTANT:
When I use knox gateway, nobody knows that I am using 50070 port internally. This way I can hide all the ports from the external world. I can also block all the port and just keep the knox gateway port (8443) open to a secure environment.

ldap user – **guest** (check *users.ldif file for more information about users used by demo-ldap)*
ldap user's password – guest-password
knox gateway host – knox.hadoop.com
gateway port – 8443
gateway path – /gateway/default
service – webHDFS (webhdfs/v1/)
operation to perform – LISTSTATUS

In Above example, we have listed the directories under **/user** using curl and knox gateway.
Output result will be displayed in JASON format. You can use any JSON formatter online, to format this json output.



3_ Write file to HDFS using Knox gatway

a) Define Variable

```
# KNOX_HOSTNAME=knox.hadoop.com
# USERNAME="op1"
# PASS="Hadoop@1"
```

b) Register the name for a sample file README in /tmp

```
# curl -i -k -u $USERNAME:$PASS -X PUT "https://$KNOX_HOSTNAME:8443/gateway/default/webhd
fs/v1/tmp/README?op=CREATE"
```

c) Upload README to /tmp directory on HDFS (hint: Use Value of Location header from command above)

```
# curl -i -k -u $USERNAME:$PASS -X PUT '{https://hdp1.c.verdant-rider-212512.internal:844
3/gateway/default/webhdfs/data/v1/webhdfs/v1/tmp/README?_=AAAACAAAABAAAADQ5LY8-eixe9bO-w8
d4seSLyCaQzLCSFmavvXrNpySilHJzY0IO9pK3UPYwdWeZlxN650S4B7iiFNtKTyvryqyGTZNYvRg8DI16PQ0e3Zy
zpp7nvEBN6F1X4P-98qoiUO3XG0OJpMR_iXr4R2hrLq_mK0r0PZnaDWw0mK7GX4a_KmQK_nVNB9j5OU6BTWKlo_jG
f7mUPbcUwepYViFopCFDn56BF_DVPF5gw9FyVHtIaqgRhd9K1kuxIFzghwVtOayJfWMedCm49BwwWb0rYkbqsb-Ql
FWlnjem1egHNTYAH8IowcXrVzS_Q}'
```

d) Verify whether file is uploaded or not

```
[root@hdp1 2.2.0]# hadoop fs -ls /tmp
Found 3 items
-rwxr-xr-x   3 op1       hdfs          0 2018-08-16 16:36 /tmp/README
drwxr-xr-x   - hdfs      hdfs          0 2018-08-06 12:46 /tmp/entity-file-history
drwx-wx-wx   - ambari-qa hdfs          0 2018-08-06 12:56 /tmp/hive
```

Setting Up LDAP/AD Authentication

LDAP authentication is configured by adding a "ShiroProvider" authentication provider to the cluster's topology file. When enabled, the Knox Gateway uses Apache Shiro (`org.apache.shiro.realm.ldap.JndiLdapRealm` or `org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm`) to authenticate users against the configured LDAP store.

Setup AD Authentication - KNOX

1) Add proxy user details for Knox in *core-site.xml*

```
hadoop.proxyuser.knox.groups=*
hadoop.proxyuser.knox.hosts=*
```

2) Configure Knox Topology for AD/LDAP Authentication (Make sure you Stop Demo LDAP if it is running)

3) Goto : Ambari > Knox > Config > Advance Topology
Use below Sample configuration to configure AD with Knox. You can refer websites given at the end of this document to know more about these properties.

We have also given a link to download this configuration in a text file. You can use any one configuration sample from below given links

Sample file 1 : https://drive.google.com/open?id=1fgLWZt1gCWtgktt5Z9laaV4_FTDy1jl0
Sample file 2 : https://drive.google.com/open?id=1QetpakWQZwLfm0AFQ6x4dpIJ0VJgDUsI

```
<topology>
      <gateway>
                <provider>
                          <role>authentication</role>
                          <name>ShiroProvider</name>
                          <enabled>true</enabled>
                          <param>
                                    <name>sessionTimeout</name>
                                    <value>30</value>
                          </param>
                          <param>
                                    <name>main.ldapRealm</name>
                                    <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</value>
                          </param>
<!-- changes for AD/user sync -->
                          <param>
                                    <name>main.ldapContextFactory</name>
                                    <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory</value>
                          </param>
                          <param>
                                    <name>main.ldapRealm.contextFactory</name>
                                    <value>$ldapContextFactory</value>
                          </param>
                          <param>
                                    <name>main.ldapRealm.contextFactory.url</name>
                                    <value>ldap://adserver.c.verdant-rider-212512.internal:389</value>
                          </param>
                          <param>
                                    <name>main.ldapRealm.contextFactory.systemUsername</name>
```

```
                        </param>
                        <param>
                                <name>main.ldapRealm.contextFactory.systemPassword</name>
                                <value>Hadoop@1</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.contextFactory.authenticationMechanism</name>
                                <value>simple</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.userDnTemplate</name>
                                <value>CN={0},OU=AppUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.searchBase</name>
                                <value>OU=AppUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.userObjectClass</name>
                                <value>person</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.userSearchAttributeName</name>
                                <value>sAMAccountName</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.authorizationEnabled</name>
                                <value>true</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.groupSearchBase</name>
                                <value>OU=AppUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.groupObjectClass</name>
                                <value>group</value>
                        </param>
                        <param>
                                <name>main.ldapRealm.groupIdAttribute</name>
                                <value>cn</value>
                        </param>
                        <param>
                                <name>urls./**</name>
                                <value>authcBasic</value>
                        </param>
                </provider>
        <provider>
            <role>authorization</role>
            <name>AclsAuthz</name>
            <enabled>true</enabled>
            <param name="knox.acl" value="*;*;*"/>
        </provider>
<!-- changes for AD/user sync -->
                <provider>
                        <role>identity-assertion</role>
                        <name>Default</name>
                        <enabled>true</enabled>
```

```
</gateway>

<!-- Services -->
      <service>
                <role>NAMENODE</role>
                <url>hdfs://{{namenode_host}}:{{namenode_rpc_port}}</url>
      </service>

      <service>
                <role>JOBTRACKER</role>
                <url>rpc://{{rm_host}}:{{jt_rpc_port}}</url>
      </service>

      <service>
                <role>WEBHDFS</role>
                {{webhdfs_service_urls}}
      </service>
</topology>
```

Note: (For Ranger)
To enable Ranger for Knox, find 'AclsAuthz' string in the above topology and replace with 'XASecurePDPKnox'.

Configuration Example:

Parameter	AD
main.ldapRealm	org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm
main.ldapContextFactory	org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory
main.ldapRealm.contextFactory	$ldapContextFactory
main.ldapRealm.contextFactory.url	ldap://adserver.c.verdant-rider-212512.internal:389
main.ldapRealm.contextFactory.systemUsername	CN=hdpadmin,OU=ServiceUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com
main.ldapRealm.contextFactory.systemPassword	Hadoop@1
main.ldapRealm.contextFactory.authenticationMechanism	simple
main.ldapRealm.userDnTemplate	cn={0},OU=AppUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com
main.ldapRealm.userSearchBase	OU=AppUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com
main.ldapRealm.userSearchAttributeName	sAMAccountName
main.ldapRealm.userObjectClass	person
main.ldapRealm.authorizationEnabled	true
main.ldapRealm.groupSearchBase	OU=AppUsers,OU=hadoop,DC=hortonworks,DC=lab,DC=com
main.ldapRealm.groupObjectClass	group
main.ldapRealm.groupIdAttribute	cn
main.ldapRealm.memberAttribute	member
main.cacheManager	org.apache.shiro.cache.ehcache.EhCacheManager
main.securityManager.cacheManager	$cacheManager
main.ldapRealm.authenticationCachingEnabled	true
urls./**	authcBasic
identity-assertion	Default

Validate:

1) Make sure Knox is configured to use CA certificates

```
# openssl s_client -showcerts -connect knoxhostname:8443
```

2) Validate Topology definition

```
# /usr/hdp/current/knox-server/bin/knoxcli.sh validate-topology --cluster default

File to be validated:
/usr/hdp/2.6.5.0-292/knox/bin/../conf/topologies/default.xml
==========================================
Topology file validated successfully
```

3) Test LDAP Authentication and Authorization

```
/usr/hdp/current/knox-server/bin/knoxcli.sh user-auth-test [--cluster c] [--u username] [
--p password] [--g] [--d]
```

```
# /usr/hdp/current/knox-server/bin/knoxcli.sh user-auth-test --cluster HortonProd --u op1
--p Hadoop@1
org.apache.hadoop.gateway.util.KnoxCLI$LDAPCommand$NoSuchTopologyException: Topology Hort
onProd does not exist in the topologies directory.

[root@hdp1 deployments]# cd /etc/knox/conf/topologies/
[root@hdp1 topologies]# ll
total 24
-rw-r--r-- 1 knox knox 1820 Aug 14 16:36 admin.xml
-rw-r--r-- 1 knox knox 3176 Aug 16 16:52 default.xml
-rw-r--r-- 1 knox knox 3307 Aug 14 16:36 knoxsso.xml
-rw-r--r-- 1 knox knox 4968 May 11 07:51 manager.xml
-rw-r--r-- 1 knox knox   89 May 11 07:51 README

[root@hdp1 topologies]# /usr/hdp/current/knox-server/bin/knoxcli.sh user-auth-test --clus
ter default --u op1 --p Hadoop@1
LDAP authentication successful!
```

4) Test the ability to connect, bind, and authenticate with the LDAP server

```
# /usr/hdp/current/knox-server/bin/knoxcli.sh system-user-auth-test --cluster default --d
System LDAP Bind successful.
```

5) List Topologies

```
[root@hdp1 topologies]# /usr/hdp/current/knox-server/bin/knoxcli.sh list-topologies
```

Access WebHDFS with & without KNOX

1) Without KNOX

```
# curl http://node2:50070/webhdfs/v1/?op=LISTSTATUS
```

2) With KNOX

```
# curl –k –u hadoopadmin:Hadoop@1 "https://knox.hadoop.com:8443/gateway/default/webhdfs/v
1/?op=LISTSTATUS"
```

```
# curl –k –u hr1:Hadoop@1 "https://knox.hadoop.com:8443/gateway/default/webhdfs/v1/user?o
p=LISTSTATUS"
```

```
# curl –k –u op1:Hadoop@1 "https://knox.hadoop.com:8443/gateway/default/webhdfs/v1/?op=LI
STSTATUS"
```

You can also use WebBrowser to execute below Knox URL.

Open Chrome and past Knox URL you want to execute. Ex:
https://knox.hadoop.com:8443/gateway/default/webhdfs/v1/?op=LISTSTATUS"

Access YARN

Access YARN through YARN API's (without Knox)

Get Cluster Information

```
curl http://resourcemanager.hadoop.com:8088/ws/v1/cluster/info
```

```
curl http://resourcemanager.hadoop.com:8088/ws/v1/cluster/apps
```

Get Cluster Metrics

```
curl http://resourcemanager.hadoop.com:8088/ws/v1/cluster/metrics
```

Get Applications Statistics

```
curl http://resourcemanager.hadoop.com:8088/ws/v1/cluster/appstatistics
```

Kill an Application

```
curl -v -X PUT -H "Content-Type: application/json"  -d '{"state": "KILLED"}' 'http://reso
urcemanager.hadoop.com:8088/ws/v1/cluster/apps/application_1534504588723_0014/state'
```

Access YARN via Knox

```
KNOXHOST=knox.hadoop.com
USERNAME=op1
PASSWD=Hadoop@1
```

```
curl -ik -u $USERNAME:$PASSWD -X GET \
"https://$KNOXHOST:8443/gateway/default/resourcemanager/v1/cluster"
```

```
curl -ik -u $USERNAME:$PASSWD -X GET https://knox.hadoop.com:8443/gateway/default/resourc
emanager/v1/cluster
```

```
curl -ik -u $USERNAME:$PASSWD -X GET https://knox.hadoop.com:8443/gateway/default/resourc
emanager/v1/cluster/info
```

```
curl -ik -u $USERNAME:$PASSWD -X GET https://knox.hadoop.com:8443/gateway/default/resourc
emanager/v1/cluster/metrics
```

```
curl -ik -u $USERNAME:$PASSWD -X GET https://knox.hadoop.com:8443/gateway/default/resourc
emanager/v1/cluster/scheduler
```

Access Hive:

Pre-requisites:

To enable Hive working with Knox, you need to change the transport mode from binary(default) to http.

```
hive.server2.transport.mode=http
```

Note:
If you do not change transport mode to HTTP, you will get connection refused error. Binary runs on port 10000, http runs on port 10001. When binary transport mode is still active Knox will try to connect to port 10001 which is not available and thus fails with "Connection refused".

Verify direct connection to hive server2 using any of the below beeline connect strings.

Direct - Binary Transport Mode

```
!connect jdbc:hive2://hiveserver2.hadoop.com:10000/;
```

Direct - HTTP Transport Mode

```
!connect jdbc:hive2://hiveserver2.hadoop.com:10001/;transportMode=http;httpPath=cliservic
e
```

ZooKeeper - Binary Transport Mode

```
!connect jdbc:hive2://zk1.hadoop.com:2181,zk2.hadoop.com:2181,zk3.hadoop.com:2181/;servic
eDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2
```

Access Hive through KNOX:

Set Variables:

Change values according to your cluster

```
KNOXHOST=knox.hadoop.com
KNOXPORT=8443
USERNAME=op1
PASSWD=Hadoop@1
KEYSTORE=/var/lib/knox/data-2.6.5.0-292/security/keystores/gateway.jks
KNOXMASTERSECRET=hadoop
```

Verify Variable values set or not:

```
echo -e "$KNOXHOST \n$KNOXPORT\n$USERNAME\n$PASSWD\n$KEYSTORE\n$KNOXMASTERSECRET"
```

Connect beeline through KNOX gateway:

Ex1:

```
beeline -n $USERNAME -p $PASSWD -u "jdbc:hive2://$KNOXHOST:$KNOXPORT/;ssl=true;sslTrustSt
ore=$KEYSTORE;trustStorePassword=$KNOXMASTERSECRET;transportMode=http;httpPath=gateway/de
fault/hive"
```

Ex2:

```
beeline  -u "jdbc:hive2://knox.hadoop.com:8443/;ssl=true;sslTrustStore=gateway.jks;trustS
```

OPTIONAL

You can also extract Knox SSL certificate and store it into a file ***knox-truststore.jks*** using java keytool.

```
cd /var/lib/knox/data-2.6.5.0-292/security/keystores/
```

```
echo "" | openssl s_client -connect $KNOXHOST:8443  2>/dev/null | sed -n -e '/BEGIN\ CERT
IFICATE/,/END\ CERTIFICATE/ p' > knox-cert.pem
```

```
keytool -importcert -alias knox-gateway -keystore knox-truststore.jks -storepass hadoop -
file knox-cert.pem
```

Assessment:

1. Once you start this exercise, go ahead and create your cluster. Login to Ambari and ensure that all services are up and running.
2. Next, install Knox service on any one of your cluster node. For this exercise you will also be interacting with Hive via Knox, so please go ahead and install Hive as well.
3. Go to Knox configuration directory (/etc/knox/conf) and check what all files/directories it contain
4. Check each files and its contents
5. Start Demo LDAP
6. Go to Knox configuration directory (/etc/knox/conf) and check what all files/directories it contain after starting knox Demo LDAP? Is there any new file or directory created?
7. View newly created files
8. Access WebHDFS with and without knox (use guest/guest-password while accessing through Knox)
9. Stop Demo LDAP
10. After this we will need to configure Knox to **authenticate with Active Directory**. Refer attached configuration file or online reference guide to configure knox with LDAP/AD and perform below Taks.
11. Execute the following HDFS operations via both cURL and browser as any AD user :
    - List Home directory
    - (if not present then) Create a home directory for your AD user in HDFS /user
    - Create a file called “file1” in /user/<username>/
    - Upload some content for “file1”
    - Read back and verify the contents of “file1”
12. Execute the following YARN operations via both cURL and browser as any AD user:
    - Get YARN cluster info/status
    - Get YARN cluster metrics
    - Get YARN Scheduler information
13. Use Beeline to connect to HS2 via Knox using the following connection string:
    - !connect jdbc:hive2://<KNOX-FQDN>:8443/;ssl=true;sslTrustStore=/home/<user>/knox-truststore.jks;trustStorePassword=hadoop;transportMode=http;httpPath=gateway/default/hive
        - You’ll need to extract Knox SSL certificate into knox-truststore.jks as follows:
            - echo "" | openssl s_client -connect <KNOX-FQDN>:8443 2>/dev/null | sed -n -e '/BEGIN\ CERTIFICATE/,/END\ CERTIFICATE/ p' > ./knox-cert.pem
            - /usr/bin/keytool -importcert -alias knox-gateway -keystore knox-truststore.jks -storepass hadoop -file ./knox-cert.pem
        - Ensure that HS2 is running in HTTP mode.
14. Once connected to HS2 via Knox, perform the following:
    - List all databases
    - List all tables
    - Run any select * query
15. Enable debug for Knox Gateway process via gateway-log4j configuration in Ambari, by setting this line:
    - log4j.logger.org.apache.hadoop.gateway=DEBUG
16. Save & Restart Knox.
17. Run any one cURL command that worked previously and observe the log file output in /var/log/knox/gateway.log
18. Run same cURL command with wrong password and now observe/compare the log file output.

Reference:

https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/knox-gateway-overview.html
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/validating_service_connectivity.html
https://github.com/kminder/knox-ldap-realm-sample
https://cwiki.apache.org/confluence/display/KNOX/Examples+WebHDFS
https://community.hortonworks.com/articles/114601/how-to-configure-and-troubleshoot-a-knox-topology.html
http://knox.apache.org/books/knox-0-12-0/user-guide.html#LDAP+Configuration
http://knox.apache.org/books/knox-0-12-0/user-guide.html#Identity+Assertion
http://knox.apache.org/books/knox-0-12-0/user-guide.html#Advanced+LDAP+configuration+precedence
http://knox.apache.org/books/knox-0-12-0/user-guide.html#LDAP+Authentication+Caching
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/setting_up_authorization_provider.html

If your cluster has external and internal hostname (eg : Cluster deployed on EC2 Instance), Refre below doc for Hostname mapping
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/knox_mapping_internal_nodes_to_external_urls.html

LDAP Configuartion
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/configuring_authentication_knox.html

Create LDAP/AD password Alias to use in configuration
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/ch02s08s06s03s04s02.html

AD Caching
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/ldap_authentication_caching.html

Example AD Configutation
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.5/bk_security/content/example_ad_configuration.html

WebUI
https://community.hortonworks.com/articles/81713/configure-knox-to-access-hdfs-ui.html
https://community.hortonworks.com/articles/106968/how-to-enable-knox-proxying-for-zeppelin.html

Identity Assertion
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.3.2/bk_Knox_Gateway_Admin_Guide/content/ch07.html

Troubleshoot Knox issues
https://community.hortonworks.com/articles/113013/how-to-troubleshoot-and-application-behind-apache.html

YARN API's
https://hadoop.apache.org/docs/r2.7.3/hadoop-yarn/hadoop-yarn-site/ResourceManagerRest.html
https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.3/bk_yarn-resource-management/content/ch_yarn_rest_apis.html

https://knox.apache.org/books/knox-1-1-0/user-guide.html#Yarn+Examples+via+cURL