Apache Knox Dynamic Configuration End-to-End

Introduction

Knox 0.14.0 includes a number of new features involving/supporting dynamic configuration.

- Simple Topology Descriptors for Topology Generation and Deployment
- Cluster service endpoint URL discovery
- Ambari Cluster Monitoring
- ZooKeeper Configuration Monitoring

This article assumes the existence of an Ambari(2.6+)-managed Hadoop cluster, and a ZooKeeper (part of the cluster, or any other). The HDP sandbox is sufficient.

This article will walk through (at a high level):

- The configuration of Knox to support these new features
- The creation of a basic provider configuration and simple descriptor
- The deployment of this descriptor (including service URL discovery)
- The use of the Admin API for managing simple descriptors and provider configurations
- The monitoring of, and response to, cluster configuration changes.
- Provider configuration and descriptors hosted in ZooKeeper
- The Knox CLI additions for managing these resources in ZooKeeper

Ambari Configuration

This step can be skipped for the HDP Sandbox, where the user ***maria_dev*** has adequate permissions.

Otherwise, create a user with the **Cluster User** role in Ambari (e.g., discovery). The **Cluster User** role can view information about the cluster and its services,
including configurations, service status, and health alerts.
This user will be employed by Knox for discovering service configuration details for a cluster, without having administrator privileges.

Gateway Configuration

The gateway interacts with the Ambari API to get details about clusters. For this purpose, credentials are required, and aliases are used primarily for passwords.

The default aliases used are:

Alias	Value
ambari.discovery.user	The username with which to connect to Ambari if no user is specified in a descriptor.
ambari.discovery.password	The password to use to connect to Ambari if no alias is specified in a descriptor.

For this example, define the **ambari.discovery.password** alias for the Knox instance prior to any attempt to deploy a simple descriptor.

```
{GATEWAY_HOME}/bin/knoxcli.sh create-alias ambari.discovery.password --value discoverysecret (for maria_dev
user, this value is maria_dev)
```

Don't worry about the username now, since it will be explicitly specified in the descriptor you deploy.

Ambari Cluster Monitoring

Knox is capable of monitoring Ambari-managed clusters, from which it has generated topologies, for configuration changes.

By default, this monitor is disabled. To enable it, the value of the following property in **gateway-site.xml** must be *true*.

```
<property>
    <name>gateway.cluster.config.monitor.ambari.enabled</name>
    <value>true</value>
    <description>Enable/disable Ambari cluster configuration monitoring.</description>
</property>
```

If this monitor is enabled, the value of the following property will determine the frequency with which it will check the Ambari configuration for changes.

```
<property>
    <name>gateway.cluster.config.monitor.ambari.interval</name>
    <value>30</value>
    <description>The interval (in seconds) for polling Ambari for cluster configuration changes.</description>
</property>
```

ZooKeeper Configuration Monitoring

Knox is capable of monitoring Apache ZooKeeper for provider configurations and descriptors. To enable this, the following properties must be defined in **ga teway-site.xml.**
(N.B., The **address** component of the **gateway.remote.config.registry.sandbox-zookeeper-client** property value must correspond to the ZooKeeper being used.)

```
<property>
    <name>gateway.remote.config.registry.sandbox-zookeeper-client</name>
    <value>type=ZooKeeper;address=localhost:2181</value>
    <description>ZooKeeper configuration registry client.</description>
</property>

<property>
    <name>gateway.remote.config.monitor.client</name>
    <value>sandbox-zookeeper-client</value>
    <description>Remote configuration monitor client name.</description>
</property>
```

Start the Gateway

If it's not already running, start the demo LDAP server: **{GATEWAY_HOME}/bin/ldap.sh start**

Start the gateway with the aforementioned configuration changes: **{GATEWAY_HOME}/bin/gateway.sh start**

Create a Provider Configuration

Provider configurations are externalized from topology descriptors to promote sharing across topologies. Their content is exactly what you would find in any topology XML prior to this release.

Create **sandbox-providers.xml** with the follwing content:

```
<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>
        <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.userDnTemplate</name>
         <value>uid={0},ou=people,dc=hadoop,dc=apache,dc=org</value>
        </param>
        <param>
         <name>main.ldapRealm.contextFactory.url</name>
         <value>ldap://localhost:33389</value>
        </param>
        <param>
         <name>main.ldapRealm.contextFactory.authenticationMechanism</name>
         <value>simple</value>
        </param>
        <param>
         <name>urls./**</name>
         <value>authcBasic</value>
        </param>
    </provider>
</gateway>
```

Create a Simple Descriptor

Simple descriptors, as the name implies, are intended to simplify the declaration of which services should be included in a topology. By including a few discovery details, Knox can determine the endpoint URLs for the declared services.
The endpoint URLs can still be specified in these descriptors, for instance, if Ambari is not managing the target cluster or you need to override the endpoint for some reason.

Create **discovery-sandbox.json** with the following content:

In this descriptor, the **discovery-user** property value needs to match the Ambari user with at least **Cluster User** permissions. For this example, it is either ***maria_dev*** (for the HDP sandbox) or whatever user
you've defined for this purpose (see Ambari Configuration).

JSON Descriptor

```
{
  "discovery-address":"http://localhost:8080",
  "discovery-user":"discovery",
  "provider-config-ref":"sandbox-providers",
  "cluster":"Sandbox",
  "services":[
    {"name":"NAMENODE"},
    {"name":"JOBTRACKER"},
    {"name":"WEBHDFS"},
    {"name":"WEBHCAT"},
    {"name":"OOZIE"},
    {"name":"WEBHBASE"},
    {"name":"RESOURCEMANAGER"}
  ]
}
```

Deploy the Configuration

These configuration files need to be deployed to the Knox instance, which will result in the generation and deployment of the **discovery-sandbox.xml** topology.
There are multiple means to accomplish this, including simple file copies (if you have access to the gateway host filesystem) and the Admin API.

File Copy

If you have access to the Knox host filesystem, you can simply copy the provider configuration and simple descriptor to the ***{GATEWAY_HOME}*****/conf /shared-providers** and ***{GATEWAY_HOME}*****/conf/descriptors** directories respectively.
For obvious reasons, it's important to deploy the provider configuration before any referencing descriptors.

```
cp sandbox-providers.xml {GATEWAY_HOME}/conf/shared-providers/
cp discovery-sandbox.json {GATEWAY_HOME}/conf/descriptors/
```

Admin API

Alternatively, the Knox Admin API can be used to deploy the configuration files

Replace these placeholders in the following URLs: ***{gateway-host}*** (e.g., localhost), ***{gateway-port}*** (e.g., 8443), ***{gateway-path}*** (e.g., gateway)

Deploy the Provider Configuration

```
curl -iku admin:admin-password https://{gateway-host}:{gateway-port}/{gateway-path}/admin/api/v1/providerconfig
/sandbox-providers \
     -X PUT \
     -H Content-Type:application/xml \
     -d "@sandbox-providers.xml"
```

Deploy the Descriptor

```
curl -iku admin:admin-password https://{gateway-host}:{gateway-port}/{gateway-path}/admin/api/v1/descriptors
/discovery-sandbox \
     -X PUT \
     -H Content-Type:application/json \
     -d "@discovery-sandbox.json"
```

Following either method of deployment, review the resulting topology file, **discovery-sandbox.xml**, and you'll find that it's a normal Knox topology, with all of the service URLs populated according to the cluster being proxied.

Ambari Cluster Monitoring

Knox is capable of monitoring Ambari instances, from which it has previously discovered cluster details, to identify changes that affect deployed topologies.

When the monitor discovers changes, it will trigger regeneration of the affected deployed topologies based on the changes, and redeploy the same.

Try It

To see this in action, try modifying a configuration property for a service in your cluster:

1. Navigate in Ambari to the **Advanced yarn-site** config, change the **yarn.http.policy** value to **HTTPS_ONLY**, and save the configuration.
2. Watch ***{GATEWAY_HOME}*/logs/gateway.log** for messages about noticing a cluster configuration change and regeneration/redeployment of the associated topology.
3. View the contents of the updated corresponding topology in ***{GATEWAY_HOME}*/conf/topologies/**, and note that the **RESOURCEMANAGER** URL now has an **https** scheme and a different port.
4. In Ambari, change the property value back to **HTTP_ONLY**, and note the subsequent update of the **RESOURCEMANAGER** URL in the re-generated topology.

ZooKeeper Configuration Monitoring

We've seen two means for deploying provider configurations and simple descriptors to a Knox instance: The host filesystem and the Admin API.

Knox is also capable of monitoring Apache ZooKeeper for the addition/modification/removal of provider configurations and simple topology descriptors, and effecting gateway changes as a result.

When a provider configuration is added or modified under the monitored **/knox/config/shared-providers** znode, every Knox instance that is monitoring this znode will download the new file to its local ***{GATEWAY_HOME}*/conf/shared-providers** directory. Note that updates to a provider configuration will result in updates to every generated topology that references it.

When a descriptor is added or modified under the monitored **/knox/config/descriptors** znode, every Knox instance that is monitoring this znode will download the new file to its local ***{GATEWAY_HOME}*/conf/descriptors** directory, and attempt to generate a toplogy based on its contents.

Removals from these znodes are treated similarly, in that they result in the removal of the corresponding local files. When a descriptor is removed, it results in the undeployment of the corresponding topology.

KnoxCLI Support for Configuration in ZooKeeper

The Knox CLI provides some commands to faciliate easier management of these configuration files in ZooKeeper:

Command	**Function**
list-registry-clients	Lists the configured registry clients available. In this case, there is likely only one: the sandbox-zookeeper-client configured for the ZooKeeper monitor.
upload-provider-config *filePath* --registry-client *name* [--entry-name *entryName*]	Uploads a provider configuration as a child of the **/knox/config/shared-providers** znode.
upload-descriptor *filePath* --registry-client *name* [--entry-name *entryName*]	Uploads a descriptor as a child of the **/knox/config/descriptors** znode.
delete-provider-config *providerConfig* --registry-client *name*	Deletes a provider configuration from the **/knox/config/shared-providers** znode.
delete-descriptor *descriptor* --registry-client *name*	Deletes a descriptor from the **/knox/config/shared-providers** znode.

Try It

Invoke the Knox CLI commands to upload the configuration files you've created to ZooKeeper:

***{GATEWAY_HOME}*/bin/knoxcli.sh** upload-provider-config sandbox-providers.xml --registry-client sandbox-zookeeper-client --entry-name alt-providers.xml

***{GATEWAY_HOME}*/bin/knoxcli.sh** upload-descriptor discovery-sandbox.json --registry-client sandbox-zookeeper-client --entry-name sandbox-copy.json

Knox will notice these additions, and the result will be **sandbox-copy.xml** in ***{GATEWAY_HOME}*/conf/topologies**.

Repeat the Ambari cluster configuration change test, and notice that both **discovery-sandbox.xml** and **sandbox-copy.xml** are regenerated and redeployed with updated **RESOURCEMANAGER** URLs.

Effect Policy Change Across Multiple Topologies

Add the **hostmap** provider to the local copy of **sandbox-providers.xml** you created earlier:

```
<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>
        <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.userDnTemplate</name>
         <value>uid={0},ou=people,dc=hadoop,dc=apache,dc=org</value>
        </param>
        <param>
         <name>main.ldapRealm.contextFactory.url</name>
         <value>ldap://localhost:33389</value>
        </param>
        <param>
         <name>main.ldapRealm.contextFactory.authenticationMechanism</name>
         <value>simple</value>
        </param>
        <param>
         <name>urls./**</name>
         <value>authcBasic</value>
        </param>
    </provider>
    <provider>
        <role>hostmap</role>
        <name>static</name>
        <enabled>true</enabled>
        <param>
          <name>localhost</name>
          <value>sandbox,sandbox.hortonworks.com</value>
        </param>
    </provider>
</gateway>
```

And replace the **sandbox-providers.xml** configuration in ZooKeeper with this updated version:

```
{GATEWAY_HOME}/bin/knoxcli.sh upload-provider-config sandbox-providers.xml --registry-client sandbox-zookeeper-
client
```

As a result, both **discovery-sandbox.xml** and **sandbox-copy.xml** are regenerated and redeployed because their descriptors reference **sandbox-providers.xml**.
The updated topologies will both include the **hostmap** provider.

Undeploy a Topology by Removing Its Corresponding Descriptor

Delete the descriptor from ZooKeeper, and the corresponding topology will be undeployed from the Knox instance:

```
{GATEWAY_HOME}/bin/knoxcli.sh delete-descriptor sandbox-copy.json --registry-client sandbox-zookeeper-client
```

Remove a Provider Configuration

Delete the provider configuration from ZooKeeper, and the corresponding file will be removed from the Knox istance.

```
{GATEWAY_HOME}/bin/knoxcli.sh delete-provider-config alt-providers.xml --registry-client sandbox-zookeeper-client
```

Summary

With these features, Knox has become easier to use. Administrators no longer need to track down all the service URLs for each and every toplogy they deploy. It's now possible to effect
policy changes for multiple topologies across multiple Knox instances with a single file change in one location. Knox has also gained the ability to dynamically respond to cluster changes.

The User Guide has more details about these new features.