# Dynamic HA Provider Configuration

## Goal

With the introduction of cluster detail discovery and topology generation in Apache Knox 0.14.0, it has become possible to make the configuration for proxying HA-enabled Hadoop services more dynamic/automatic.

Furthermore, it may even be possible for Knox to recognize the HA-enabled configuration for a service, and automatically configure a topology to interact with that service in an HA manner.

Knox could also leverage the Ambari cluster monitoring capability to respond to cluster changes, whereby service configurations are modified to enable HA, by regenerating and redeploying the affected topologies.

## Current HA Configuration

Currently, topology services can accommodate proxying the corresponding HA-enabled cluster services using topology configuration or by leveraging ZooKeeper; the means depends on the cluster service.
For example, the WEBHDFS service is configured with URLs in the service declaration while the HIVE service employs the list of hosts in the configured ZooKeeper ensemble:

**HA Service Configurations Topology Excerpt**

```
...
  <provider>
    <role>ha</role>
    <name>HaProvider</name>
    <enabled>true</enabled>
    <param>
      <name>WEBHDFS</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000;enabled=true</value>
    </param>
    <param>
      <name>HIVE</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;enabled=true;zookeeperEnsemble=host1:2181,host2:2181,
host3:2181;zookeeperNamespace=hiveserver2</value>
    </param>
  </provider>
</gateway>

<service>
  <role>WEBHDFS</role>
  <url>http://host1:50070/webhdfs</url>
  <url>http://host2:50070/webhdfs</url>
</service>

<service>
  <role>HIVE</role>
</service>
...
```

## Topology-Based HA

Knox already handles the topology-based HA configuration to a degree. At topology generation time, if there are multiple hosts in the cluster for the associated service, Knox will add the URLs to the **<service/>** element.
If there is also an entry for that service in the HaProvider configuration, then the service will be proxied in an HA manner.

## ZooKeeper-Based HA

Using Ambari, Knox can actually determine the ZooKeeper configuration for each service dynamically, relieving the administrator from having to explicitly configure this in each topology.

These are some examples of service configurations for which there is ZooKeeper-related information:

| Service Config | Property | Example Value |
|---|---|---|
| HIVE:hive-site | hive.zookeeper.quorum | host1:2181,host2:2181,host3:2181 |
| | hive.server2.zookeeper.namespace | hiveserver2 |
| | hive.server2.support.dynamic.service.discovery | true |
| HBASE:hbase-site | hbase.zookeeper.quorum | host1,host2,host3 |
| | hbase.zookeeper.property.clientPort | 2181 |
| | zookeeper.znode.parent | /hbase-unsecure |
| KAFKA:kafka-broker | zookeeper.connect | sandbox.hortonworks.com:2181 |
| HDFS:hdfs-site | ha.zookeeper.quorum | host1:2181,host2:2181,host3:2181 |
| | dfs.ha.automatic-failover.enabled | true (only for "Auto HA") |
| OOZIE:oozie-site | oozie.zookeeper.connection.string | localhost:2181 |
| | oozie.zookeeper.namespace | oozie |
| YARN:yarn-site | `yarn.resourcemanager.ha.enabled` | true |
| | `yarn.resourcemanager.zk-address` | host1:2181,host2:2181,host3:2181 |
| WEBHCAT:webhcat-site | templeton.zookeeper.hosts | host1:2181,host2:2181,host3:2181 |
| ATLAS:application-properties | atlas.kafka.zookeeper.connect | host1:2181,host2:2181,host3:2181 |
| | atlas.server.ha.zookeeper.connect | host1:2181,host2:2181,host3:2181 |
| | atlas.server.ha.zookeeper.zkroot | /apache_atlas |
| | atlas.server.ha.enabled | true |

# Required Changes

- Knox will parse referenced provider configurations, rather than just blindly copying their contents.
  This will serve multiple purposes:
  - It will provide a degree of validation (invalid provider configurations would fail parsing)
  - It will allow a finer level of control over what is serialized to the generated topology file:
    - Knox will have the opportunity to add cluster-specific details into provider configuration in the generated topologies
    - Knox could add/modify the HaProvider config based on cluster details (e.g., HA enabled for a particular service)
    - Knox could reference elements of the provider configuration (e.g., HaProvider) to inform the generation of service elements in the generated topologies.

- Move cluster-specific service HA configuration from the HaProvider configuration to the service declaration.
  - This just makes more logical sense
  - Moving cluster-specific details out of the provider configuration will make shared provider configurations applicable to more topologies (i.e., more reusable), across clusters.
  - The cluster-specific configuration could be discovered along with the service URL details, rather than having to be hard-coded in a provider configuration.
  - This will be treated as override configuration, so that the existing approach (i.e., complete configuration as HaProvider param values) will continue to work, to satisfy backward-compatibility requirements.

- Support new enabled value "auto" to allow service-specific configuration to determine whether HA treatment is enabled for that service.

**Example HaProvider Configuration with enabled=auto**

```
  <provider>
    <role>ha</role>
    <name>HaProvider</name>
    <enabled>true</enabled>
    <param>
      <name>WEBHDFS</name>
      <value>maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000;enabled=auto<
/value>
    </param>
  </provider>
```

In this case, the topology generation will reference the dfs.ha.automatic-failover.enabled property in the HDFS configuration to determine whether the HaProvider should be enabled for WEBHDFS.

# Open Items

- Identify the nature of all the supported services
  - Which services are currently purely topology-based (DefaultURLManager)?
    - WEBHCAT
    - OOZIE
    - WEBHDFS
    - ?
  - Which services are currently ZooKeeper-based (BaseZooKeeperURLManager)?
    - HIVE (HS2ZooKeeperURLManager)
    - HBASE (HBaseZooKeeperURLManager)
    - Kafka (KafkaZooKeeperURLManager)
    - SOLR (SOLRZooKeeperURLManager)
    - ATLAS (AtlasZookeeperURLManager)
  - Could ZooKeeper support be added for any services which do no currently support it?
    - These have zookeeper-related configuration (see table):
      - WEBHDFS
      - OOZIE
      - YARN
      - WEBHCAT
      - RESOURCEMANAGER
- For the ZooKeeper-based-HA services, determine if the ZooKeeper details are available from the service's configuration via Ambari.
- Can "HA mode" be determined for every service type from the cluster configuration details? Can Knox dynamically identify HA-configured services, and generate the topology accordingly?
- Determine how to leverage the cluster discovery data to generate the ZooKeeper HA configuration for the relevant declared topology services.

## Proposed Alternative Provider Configuration File Formats

### JSON

**JSON Provider Configuration**

```
{
  "providers": [
    {
      "role":"authentication",
      "name":"ShiroProvider",
      "enabled":"true",
      "params":{
        "sessionTimeout":"30",
        "main.ldapRealm":"org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm",
        "main.ldapContextFactory":"org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory",
        "main.ldapRealm.contextFactory":"$ldapContextFactory",
        "main.ldapRealm.userDnTemplate":"uid={0},ou=people,dc=hadoop,dc=apache,dc=org",
            "main.ldapRealm.contextFactory.url":"ldap://localhost:33389",
            "main.ldapRealm.contextFactory.authenticationMechanism":"simple",
            "urls./**":"authcBasic"
          }
    },
        {
          "role":"hostmap",
          "name":"static",
          "enabled":"true",
          "params":{
            "localhost":"sandbox,sandbox.hortonworks.com"
          }
        },
        {
          "role":"ha",
          "name":"HaProvider",
          "enabled":"true",
          "params":{
            "WEBHDFS":"maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000;
enabled=true",
                 "HIVE":"maxFailoverAttempts=3;failoverSleep=1000;enabled=true"
          }
        }
  ]
}
```

## YAML

**YAML Provider Configuration**

```
---
providers:
  - role: authentication
    name: ShiroProvider
    enabled: true
    params:
      sessionTimeout: 30
      main.ldapRealm: org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm
      main.ldapContextFactory: org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory
      main.ldapRealm.contextFactory: $ldapContextFactory
      main.ldapRealm.userDnTemplate: uid={0},ou=people,dc=hadoop,dc=apache,dc=org
      main.ldapRealm.contextFactory.url: ldap://localhost:33389
      main.ldapRealm.contextFactory.authenticationMechanism: simple
      urls./**: authcBasic
  - role: hostmap
    name: static
    enabled: true
    params:
      localhost: sandbox,sandbox.hortonworks.com
  - role: ha
    name: HaProvider
    enabled: true
    params:           # No cluster-specific details here
      WEBHDFS: maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000;enabled=true
      HIVE: maxFailoverAttempts=3;failoverSleep=1000;enabled=true
```

**Proposed Simple Descriptor Service Params - YAML**

```
---
discovery-address: http://localhost:8080
discovery-user: maria_dev
provider-config-ref: sandbox-providers
cluster: Sandbox

services:
    - name: NAMENODE
    - name: JOBTRACKER
    - name: WEBHDFS
      params:
        maxFailoverAttempts: 5
        maxRetryAttempts: 5
        failoverSleep: 1001
    - name: WEBHBASE
    - name: HIVE
      params:
        haEnabled: true
        maxFailoverAttempts: 5
        maxRetryAttempts: 5
        failoverSleep: 1001
        zookeeperNamespace: hiveserver2                                          # Optionally, omit this, and
Knox could discover it
        zookeeperEnsemble: http://host1:2181,http://host2:2181,http://host3:2181  # Optionally, omit this, and
Knox could discover it
    - name: RESOURCEMANAGER
```

**Proposed Simple Descriptor Service Params - JSON**

```
{
  "discovery-address":"http://localhost:8080",
  "discovery-user":"maria_dev",
  "provider-config-ref":"sandbox-providers",
  "cluster":"Sandbox",
  "services":[
    {"name":"NAMENODE"},
    {"name":"JOBTRACKER"},
    {"name":"WEBHDFS",
       "params": {
        "maxFailoverAttempts": "5",
        "maxRetryAttempts": "5",
        "failoverSleep": "1001"
       }
    },
    {"name":"WEBHBASE"},
    {"name":"HIVE",
       "params": {
        "haEnabled": "true",
        "maxFailoverAttempts": "4",
        "maxRetryAttempts": "6",
        "failoverSleep": "5000",
        "zookeeperNamespace": "hiveserver2",
        "zookeeperEnsemble": "http://host1:2181,http://host2:2181,http://host3:2181"
       }
    },
    {"name":"RESOURCEMANAGER"}
  ]
}
```

**Proposed Generated Topology XML**

```
...
  <provider>
    <role>ha</role>
    <name>HaProvider</name>
    <enabled>true</enabled>
    <param>
      <name>WEBHDFS</name>
      <!-- No cluster-specific details here -->
      <value>maxFailoverAttempts=3;failoverSleep=1000;maxRetryAttempts=300;retrySleep=1000;enabled=true</value>
    </param>
    <param>
      <name>HIVE</name>
      <!-- No cluster-specific details here -->
      <value>maxFailoverAttempts=3;failoverSleep=1000;enabled=true</value>
    </param>
  </provider>
</gateway>

<service>
  <role>WEBHDFS</role>
  <url>http://host1:50070/webhdfs</url>
  <url>http://host2:50070/webhdfs</url>
</service>

<service>
  <role>HIVE</role>
  <!-- Cluster-specific details here -->
  <param>
    <name>zookeeperEnsemble</name>
    <value>host1:2181,host2:2181,host3:2181</value>
  </param>
  <param>
    <name>zookeeperNamespace</name>
    <value>hiveserver2</value>
  </param>
  <param>
    <name>haEnabled</name>
    <value>true</value>
  </param>



</service>
...
```