Multi-Node Topologies

- Overview
- Basic Attivio Node
- Multi-node Deployment Options
    - Option 1: Clustered - External Hadoop
    - Option 2: Clustered - Attivio Cluster
- Multiple Environments
- Connectors, Ingestion, Query and Response Workflows
- Index Writers and Searchers
    - ZooKeeper and Hadoop Services
- A Three Node Cluster Example to Get Us Started
- Implementing a Topology
    - Create the Project
    - Modify the Configuration Files
        - Modifications to topology-nodes.xml
            - Shared Configuration
            - Configure ZooKeepers
            - Configure Attivio Nodes
            - Configure Nodesets
            - Configure Store (single-node only)
            - Configure Performance Monitor Service
            - Option 2: Clustered - Attivio Cluster Only
                - HDFS
                - HBase
                - YARN
                - Sample Topologies
            - Option 1 - External Cluster
            - Option 2 - Attivio Cluster (Minimal)
            - Option 2: Attivio Cluster (Extended)
        - Modifications to Index.index.xml
        - Modifications to BusinessCenter.xml
        - Index Writer/Searcher Considerations and Cluster Feasibility
        - Layout
            - Default Nodesets
            - Service, Connector and Workflow Locations
        - Distribution Schemes
    - Using the Agent and CLI
        - Starting AIE-Agent
        - Starting AIE-CLI
        - Running the Project
        - Monitoring the Project
        - Monitoring the ConfigServer
        - Checking Stderr and Stdout Files
        - Stopping the Project
    - Multi-Node Administration
- Backup and Restore

Overview

For efficiency, scaling, and uninterrupted query service, multiple Attivio processes can work together to implement a multi-node (or clustered) topology. With a multi-node topology, you can configure one or more nodes dedicated to content ingestion, query and response processing and can partition very large indexes across multiple processes on multiple servers to fit project needs. Topology modeling allows configuration of such systems in a simplified, unified manner.

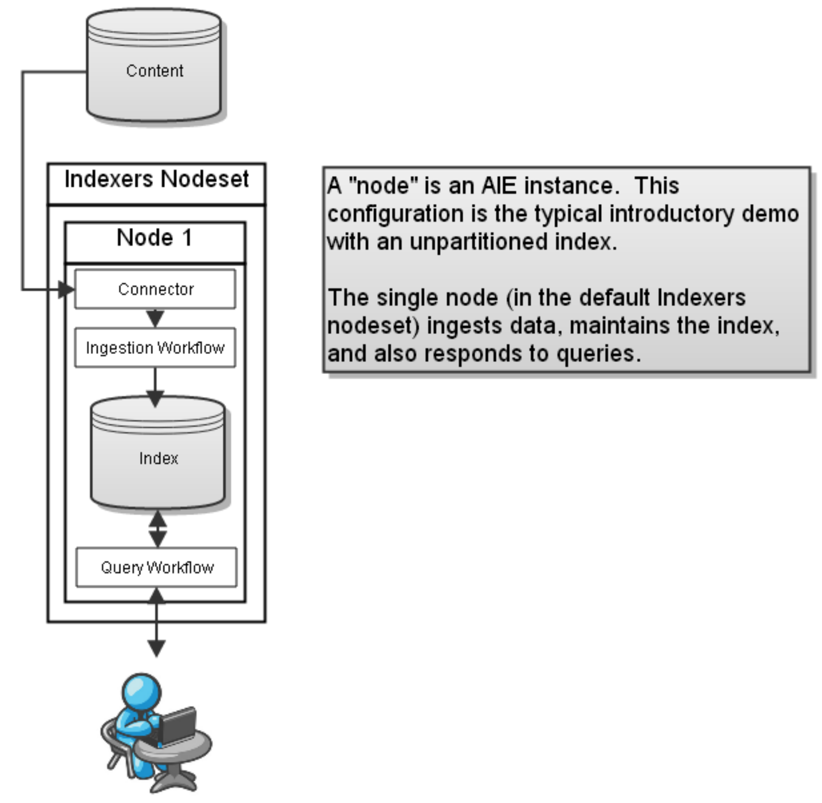
This page contains instructions for setting up multi-node topologies in Attivio. It also contains links to administration topics for the resulting system. Use the content on this page to help you make design and scaling decisions to fit your application.

See also...

See Configure the Index to view Attivio's full range of index-configuration features.

Basic Attivio Node

A node is an Attivio instance. Your early training exercises with Attivio likely used a single-node system like the one pictured here.


Content
Indexers Nodeset
Node 1
Connector
Ingestion Workflow
Index
Query Workflow
A "node" is an AIE instance. This configuration is the typical introductory demo with an unpartitioned index.
The single node (in the default Indexers nodeset) ingests data, maintains the index, and also responds to queries.

The single-node Attivio installation runs on a single computer. It ingests content through a connector, and processes the content into a single index. Queries are directed to the node, where they are processed and responded to.

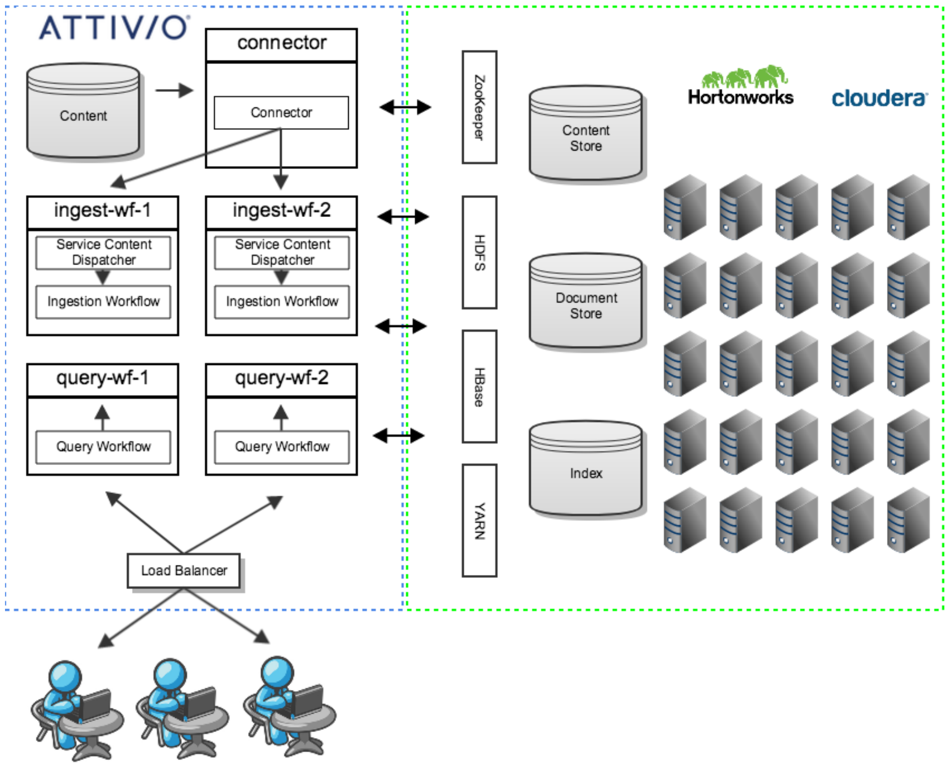
From this basic node, you can extend the system to make it more capable and robust.

Multi-node Deployment Options

Multi-node (or clustered) topologies have two deployment options:

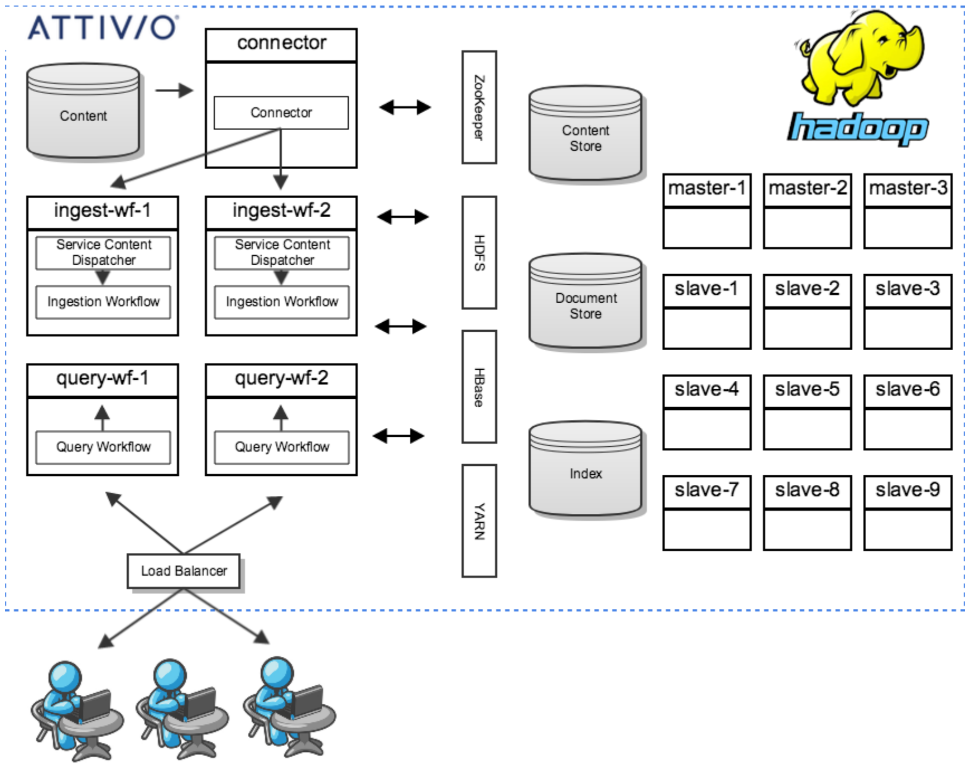
Option 1: Clustered - External Hadoop

One or more Attivio nodes handle running connectors, performance monitor and ingestion, query and response workflows while an external Hadoop cluster hosts the configuration servers, index writers and searchers and the content and document stores. Choose this option when your organization has either a Cloudera or Hortonworks cluster on which you wish to run Attivio.



Option 2: Clustered - Attivio Cluster

One or more Attivio nodes handle running connectors, performance monitor and ingestion, query and response workflows while additional Attivio-controlled nodes form a cluster which hosts the configuration servers, index writers and searchers and the content and document stores. Choose this option if your organization does not have either a Cloudera or Hortonworks cluster. Attivio will create one for you.



Multiple Environments

It is possible to create multiple topologies for different purposes in an Attivio configuration. For example, you could deploy:

- A development topology on a single machine
- A UAT topology on three servers
- A production system on three or more servers depending on your project's needs

Because the topology configuration is separate, each of these configurations can share the overall system configuration.

See Managing Environments for more information on configuring multiple environments.

Connectors, Ingestion, Query and Response Workflows

Regardless of the multi-node deployment option you choose from above, your project will require Attivio nodes to run connectors as well as ingestion, query and response workflows. While these all can run on a single node, and it may make sense to do so for a local developer workstation or test environment, for larger projects and those requiring high availability (HA), Attivio recommends configuring separate nodes for each of these activities.

With the index writers and searchers as well as the content and document stores running in Hadoop, the resources required for the nodes hosting connectors and the various workflows are reduced compared to previous versions of Attivio. This separation is designed to allow each component of an Attivio solution to scale independently. You can add connector nodes if you have an increase in the number of sources from which you want to ingest content. You can add ingestion workflow nodes if you want to do additional text analytics or natural language processing or even add a module to do OCR text extraction from your image files. Finally, you can add additional query/response workflow nodes to handle increases in query load from your application.

Index Writers and Searchers

Just as you can scale the number of nodes hosting connectors and ingestion and query workflows, you can likewise scale the number of index writers and searchers to meet increased index size and query requirements. In previous versions of Attivio, ingestion and query capacity was increased by partitioning the index, essentially splitting it into portions where the portions were spread across additional Attivio nodes.

To support high availability, higher query volume, or both, additional *rows* of searchers can be added. Each row of searchers functions as an independent query processing mini-system. To support larger content volumes, additional *columns* of index writers and searchers can be added to house subsets of partitions. These considerations are described in greater detail in Configure the Index.

Option 1: When deploying to an external Hadoop cluster, Attivio leverages the cluster's pool of resources to scale a project's index. If your ingestion or query needs change, you can re-provision the resources from the Hadoop cluster to fit your new requirements, either adding more or freeing up resources for someone else to use.

When using an external cluster and using the default high availability compatible `sliderPlacementPolicy` index property value, you must set the following property in the `<PROJECT_DIR>/conf/properties/attivio.core-app.properties` file:

Property	Value(s)
yarn.aa.placement.racks	A comma-separated list (e.g., **`/bostondc,/seattledc,/tokyodc`**) of the rack names configured for your external Hadoop cluster nodes on which index writers and searchers are allowed to run. Each rack name must include the leading slash. You can find the names of your Hadoop cluster's racks in the Cloudera Manager UI; from the main page, select the **Hosts > All Hosts** menu entry, then expand the **Rack** entry in the left-hand **Filters** menu to see a list of rack names and the number of Hadoop nodes in each rack. Cloudera CDH creates a single rack named **`/default`** if no custom rack names are configured.

This is due to a limitation with external clusters whereby the rack names are not accessible to all processes (in particular, with CDH, they appear to not be known to NodeManager processes, but are known to ResourceManager processes). For high availability compatible placement, the resource requests must include the allowed rack names.

Option 2: When choosing the Attivio cluster deployment option, the pool of resources available for the index are managed by Attivio and run on nodes within your topology. Because Attivio uses consensus-based protocols, a minimum of three nodes is required. The underlying technologies used by Attivio's cluster implementation include ZooKeeper and Hadoop. Only minimal knowledge of the specifics (primarily memory requirements and topology constraints) is required, though we provide information of primary interest for advanced use cases below.

With both cluster options, a Slider component manages the allocation of containers within the cluster for each index writer or index searcher. To correctly support high availability, as of 5.6.1, the Slider will enforce a placement policy that prohibits two containers of the same index from being allocated onto the same host. (Otherwise, for example, all rows of partition 1 for an index might be placed on the same host, thereby any outage of that host will render a search outage.) If you do *not* desire this behavior, you must explicitly set the value of index@sliderPlacementPolicy to 2 as described in Configure the Index.

ZooKeeper and Hadoop Services

In Option 1, ZooKeeper is used as a configuration service. In Option 2, ZooKeeper is used both as a configuration service and as a common service for Hadoop services. A brief overview of the various Hadoop services is provided below along with their default memory configurations under Option 2. (Under Option 1, their default memory configurations are provided by the external cluster.) In general, you do not need to know much about the Hadoop services beyond their general function and their quantity and memory requirements. (The memory requirements are needed to help size out a cluster.) HDFS is the distributed file system where an Attivio cluster stores the index data. It is also where HBASE stores its data. YARN is a compute resource allocation system to assign containers on hosts with available capacity. HBASE is a distributed NoSQL database used for the Attivio store.

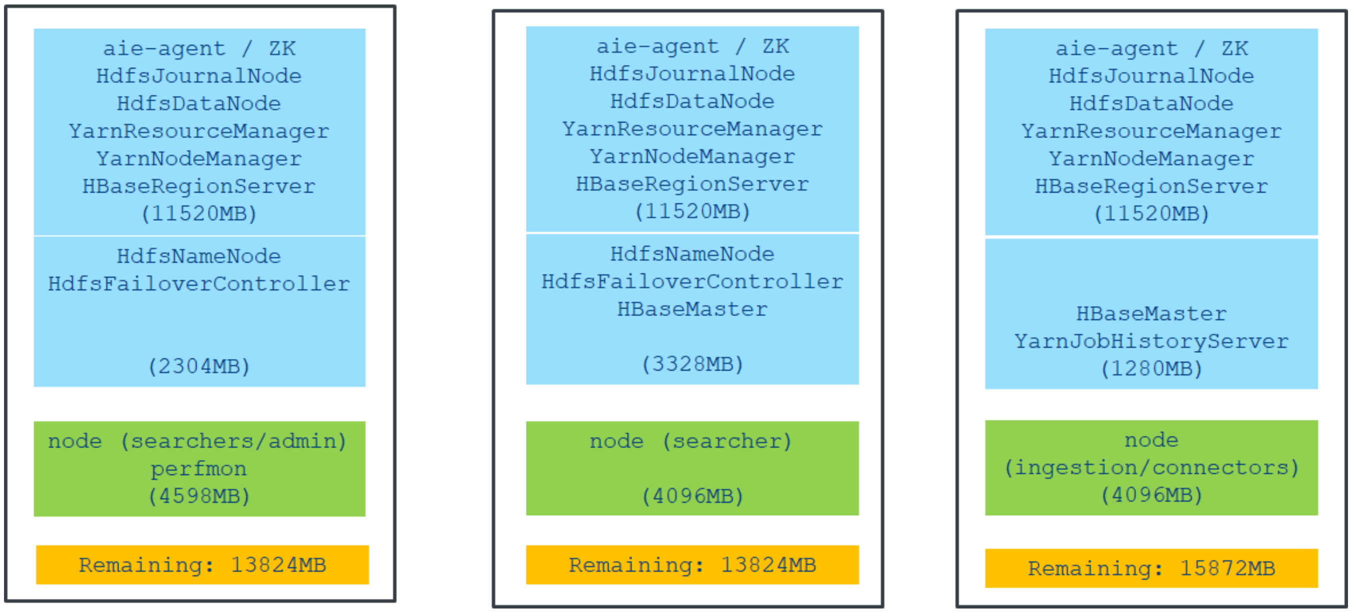
While under Option 1, Hadoop configurations are managed in the external cluster, the property **`yarn.nodemanager.pmem-check-enabled`** should be set to **`false`** . The reason for this is that the Attivio index writers/searchers run within YARN containers are Java processes and Java processes typically exceed their configured heap size leading up to a garbage collection event. If this property were set to **`true`** , those processes may be terminated by the YARN node manager, leading to instability. If cluster policies do not permit setting this property to **`false`** , you can attempt to set the **`index.process.overhead`** property in **`attivio.core-app.properties`** to a value which is the percentage overhead for indexer Java heap memory allocation. Providing for an overhead percentage (e.g. 40) would mean that 40% of the memory will not be available to indexer but will be reserved for temporary memory spikes. *It is strongly recommended that pmem check be disabled.*

Summary of services and requirements:

1. ZooKeeper is part of aie-agent. A minimum of 3 is required for HA (2 operational). 256MB each.
2. HdfsJournalNode provides transaction logging for HDFS. A minimum of 3 is required for HA (2 operational, except all 3 must be operational at initial setup). 2048MB each.
3. HdfsNameNode provides directory information for HDFS. A minimum of 2 is required for HA (1 operational). 2048MB each.
4. HdfsFailoverController monitors and elects the active HdfsNameNode. One should co-exist with each HdfsNameNode. 256MB each.
5. HdfsDataNode manages the storage of files – one should exist on each node that forms a part of the entire distributed file system. A minimum of two are required for HA. 4096MB each.
6. YarnResourceManager allocates containers to available compute hosts. A minimum of 2 is required for HA (1 operational). 1024 MB each.
7. YarnNodeManager manages containers running on a given compute host. One should exist on each node that provides compute capacity for the index writers/searchers. 1024MB each for the process itself (additional memory is configured for the containers themselves as described below).
8. YarnJobHistoryController manages history information for MapReduce jobs. One is required. 256MB each.
9. HBaseMaster assigns each HBASE table to an available HBaseRegionServer. A minimum of 2 is required for HA (1 operational). 1024MB each.
10. HBaseRegionServer services HBASE table queries. A minimum of 2 is required for HA (1 operational). 3072MB each.

A Three Node Cluster Example to Get Us Started

Below is an illustration of a three node cluster. A three node cluster can provide either high availability (via 2 rows) or higher content capacity (via 3 columns). This is the simplest cluster possible. This example is also shown with 32GB memory nodes. Using 16GB memory nodes is also possible but would require more than 3 nodes.



Some things to note about this minimal cluster example:

- Memory for infrastructure Hadoop services are under 15GB per node
- Allow 1GB for the OS
- Allow 2.5GB for abc-index
- This leaves 10GB available for the main index (either for 2 rows HA support or 3 columns larger data set support)

Implementing a Topology

Create the Project

Implementation of the topology requires both short-term and long-term planning, since the topology of evaluation systems, development systems, testing systems and deployed systems naturally vary, usually as a function of the amount of data ingested.

Windows - Unclustered Only

Unlike previous versions of Attivio, the Windows operating system is only supported for unclustered environments. Multi-node environments require the Linux operating system. It is common for developers to set up their development workstation as a separate, unclustered Attivio environment, very often using the Windows operating system.

Use the createproject tool to generate a project. Be sure to include all of the modules required to support your goals.

For example, the Quick Start exercise creates a factbook project by using createproject with the following arguments:

```
<install-dir>\bin\createproject -n Factbook -g demo -m factbook -o <attivio-projects-dir>
```

where `<attivio-projects-dir>` is a directory path such as `C:\attivio-projects`.

Modify the Configuration Files

This creates a *project directory*, which in this case is **`C:\attivio-projects\Factbook`** . In the discussions that follow, this is referred to generically as the **`<project-dir>`** . Setting up a multi-node topology is mostly a matter of editing various XML files found in the `<project-dir>\conf\` directory tree.

The **`topology-nodes.xml`** file created by the createproject tool defines a single environment called "default". Let's examine the **`<project-dir>\conf\environments\default\topology-nodes.xml`** file which is generated:

```
<attivio xmlns="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.attivio.com/configuration/config classpath:/xsd/config.xsd">
  <topology agentPort="16999" projectdir=".">
    <zookeepers>
      <zookeeper baseport="16980"/>
    </zookeepers>
    <hdfs/>
    <hbase/>
    <yarn/>
    <store baseport="16970" gc="CMS"/>
    <perfmonserver baseport="16960"/>
    <nodes>
      <node name="local" host="localhost" baseport="17000"/>
    </nodes>
    <nodesets>
      <nodeset name="admin">
        <include name="local"/>
      </nodeset>
      <nodeset name="connectors">
        <include name="local"/>
      </nodeset>
      <nodeset name="searchers">
        <include name="local"/>
      </nodeset>
      <nodeset name="ingestion">
        <include name="local"/>
        <jvmArgument value="-XX:MaxGCPauseMillis=30000"/>
      </nodeset>
    </nodesets>
  </topology>
</attivio>
```

The topology above defines a single node named "local" which will run on localhost using port **`17000`** as the base port:

```
    <nodes>
      <node name="local" host="localhost" baseport="17000"/>
    </nodes>
```

Modifications to topology-nodes.xml

The `topology-nodes.xml` file lets us configure the important aspects of the Attivio topology.

- Attivio nodes
- Nodesets
- Configuration Server(s)
- Store
- Performance Monitor service
- HDFS, HBase and YARN services (Option 2 - Attivio Cluster only)

Shared Configuration

You can use the following attributes and elements wherever you configure Attivio processes.

You cannot use properties (such as **`${data.dir}`**) for values of these attributes, but you can configure separate topology-nodes.xml files for each of your project's environments to allow use of environment-specific hosts, ports, paths, and other settings.

Note that attributes such as **`maxmem`** can be set on the **`<topology>`** element, the **`<nodeset>`** element, or the **`<node>`** element. Values are defaults that transparently inherit from the more-general elements to the more-specific ones. As a rule of thumb, the most-specific setting has priority. This lets you set topology-wide defaults but still tweak the properties of individual nodesets and nodes.

If a node inherits conflicting values due to participation in multiple nodesets, the first-defined nodeset takes precedence.

Attribute	Default	Description
debugport	*none*	If specified, the process starts with java debugging enabled.
debugSuspendOnStartup	false	If specified along with debugport, the process will be suspended on startup to enable debugging attachment before the process proceeds.
maxmem	*none, java default is used*	Maximum memory for the process, accepts numeric values followed by a 'g' for gigabytes or 'm' for megabytes.
baseport	17000	The baseport for the process.
datadir	.	The data directory for the process. Relative paths are relative to the working directory established with the agent.
gc	STD	The garbage collector to use. Allowed values are STD (default java GC), CMS, G1, or NONE. If NONE is specified, providing detailed GC settings is recommended. (See below)
host	*none*	The host where this process executes
logdir	.	The log directory for the process. Relative paths are relative to the working directory established with the agent.
projectdir	*none*	The project directory for the process. Data and log directories default to this directory.

Element	Attributes	Description
jvmArgument	value	Provides a way to add arbitrary jvm arguments to the process, e.g. `<jvmArgument value="-XX:PermSize=500"/>`
jvmProperty	name,value	Provides a way to define arbitrary jvm properties to the process, e.g. `<jvmProperty name="myprop" value="myvalue"/>` becomes `-Dmyprop=myvalue`

Configure ZooKeepers

Attivio coordinates the content of multi-node systems through one or more ZooKeeper instances (known as configuration servers in previous versions of Attivio). Configuration servers monitor the status of the Attivio project nodes and coordinate changes in the Attivio project's configuration, keeping all nodes in harmony.

A configuration server is a stand-alone application, independent of Attivio. You can start it on any host where Attivio is installed, whether an Attivio node is running on that host or not.

For a single-node, local Attivio project, the configuration server is configured in **`<project-dir>\conf\environments\default\topology-nodes.xml.`** The host attribute is not required as it is assumed to be running on the one node.

```
    <zookeepers>
      <zookeeper baseport="16980"/>
    </zookeepers>
```

Any ZooKeeper property can be configured, however, usually only one makes sense.

Property	Default	Description
dataLogDir	(none)	The location where all the ZooKeeper write-ahead log files are stored. These logs protect the integrity of the ZooKeeper data store while the state information is being updated. ZooKeeper requires that the log be flushed successfully to disk before sending a response to each request. If not set (the default), these logs will be written within the zoo subdirectory in the aie-agent's data directory. This setting typically does not need to be changed except when very slow performance (more than approx 10 seconds per write) is experienced. The cases where we have seen this is when a common SAN is used for the data directory and the SAN cannot be configured to provide higher priority to small requests. Such slow ZooKeeper WAL performance can lead to system instability. In those cases, it is recommended that a local disk, or perhaps an alternate SAN mount with provisioned IOPS, be used for the dataLogDir.

Options 1 and 2: In a clustered (multi-node) project, Attivio leverages the ZooKeeper instance in the Hadoop cluster and the list of configuration servers is specified using the **`zookeepers`** element in **`topology-nodes.xml`** . A highly available production system requires *at least three configuration servers* t o provide fault-tolerant recovery. With Option 1, the specified ZooKeepers are used only for non-Hadoop Attivio configurations. With Option 2, the ZooKeepers are used for both the included Hadoop services as well as the non-Hadoop Attivio services.

It is possible to specify ZooKeeper configuration properties both on a per node basis and for all ZooKeeper nodes. The example below shows setting the dataLogDir property for all nodes.

```
    <zookeepers>
      <zookeeper host="host1" baseport="16980"/>
      <zookeeper host="host2" baseport="16980"/>
      <zookeeper host="host3" baseport="16980"/>
      <properties>
        <property>
          <name>dataLogDir</name> <!-- optional: specify a faster disk for ZooKepper write-ahead logs -->
          <value>/mnt/fastdisk/zoo</value>
        </property>
      </properties>
    </zookeepers>
```

Configure Attivio Nodes

Edit the `<nodes>` element to specify each node in your topology. Here we specify 5 nodes.

```
<nodes>
   <node name="node1" host="host1" baseport="17000" />
   <node name="node2" host="host2" baseport="17000" />
   <node name="node3" host="host3" baseport="17000" />
   <node name="node4" host="host4" baseport="17000" />
   <node name="node5" host="host5" baseport="17000" />
</nodes>
```

The `host` value is the Domain Name System (DNS) hostname of a computer. (Note that the Configuration Server copies `topology-nodes.xml` to every host computer, so it isn't appropriate to use "localhost" when defining a node except in a single-node project.)

Attivio automatically allocates several ports (17001, 17002, 17003...) incrementing from the base port number of each node. For this reason, nodes on the same computer need a little space between their base port numbers.

Configure Nodesets

A nodeset is a list of Attivio nodes that share some common role. (Note that a single node is represented internally as a nodeset of one element, so you can use a node name as the value of a nodeset parameter.)

The createproject tool supplies nodeset definitions for four standard nodesets: **`admin`, `connectors`, `ingestion`, and `searchers`** . Following is a modified nodesets element to specify the role of each of our 5 nodes:

```
     <nodesets>
      <nodeset name="admin">
        <include name="node1"/>
      </nodeset>
      <nodeset name="connectors">
        <include name="node1"/>
        <include name="node3"/>
      </nodeset>
      <nodeset name="searchers">
        <include name="node4"/>
        <include name="node5"/>
      </nodeset>
      <nodeset name="ingestion">
        <include name="node2"/>
        <include name="node3"/>
        <jvmArgument value="-XX:MaxGCPauseMillis=30000"/>
      </nodeset>
    </nodesets>
```

Note that the **`admin`** nodeset can contain only one node.

The **`ingestion`** nodeset contains two nodes that will perform ingestion tasks. **Searchers** allows two nodes to service queries in this project. By having two nodes in the **`connectors`** nodeset, the connectors in our project will be able to be run on either of them.

Configure Store (single-node only)

The Attivio Store contains non-index data associated with the state of the current project, such as the Document Store. The purpose of the **`aie-store`** pr ocess differs depending on whether a clustered system is in use. See Architecture pages for details.

```
    <store baseport="16970" gc="CMS"/>
```

Options 1 and 2: In a clustered (multi-node) project, the Store is hosted by HBase. See the HBase section below. When using either of these options, the store element should be removed.

Configure Performance Monitor Service

The Performance Monitoring service collects diagnostic information from all nodes of the Attivio system and displays the incoming data streams as tables and graphs in the Attivio Administrator. Only one Performance Monitor service should be defined in the topology.

```
    <perfmonserver baseport="16960"/>
```

Option 2: Clustered - Attivio Cluster Only

HDFS

Specify the nodes where the HdfsNameNode, HdfsFailoverController, HdfsJournalNode, and HdfsDataNode services should run. In this example, nodes 6, 7 and 8 will be the Hadoop Master nodes and nodes 9 - 13 will be Hadoop Slave nodes.

```
    <hdfs>
      <namenode host="host6"/>
      <namenode host="host7"/>
      <failovercontrollernode host="host6"/>
      <failovercontrollernode host="host7"/>
      <journalnode host="host6"/>
      <journalnode host="host7"/>
      <journalnode host="host8"/>
      <datanode host="host9"/>
      <datanode host="host10"/>
      <datanode host="host11"/>
      <datanode host="host12"/>
      <datanode host="host13"/>
    </hdfs>
```

The following properties can be configured for HDFS, overriding their defaults.

Property	Default	Description
attivio.hadoop.data.dir	data-agent /hadoop-data *	The location where all Hadoop services will store data on the Hadoop master and slave nodes.
attivio.hdfs.namenode.memory.max-mb	2048	The maximum allocation for the HdfsNameNode services, in MBs. The minimum defaults to 1024MB.
attivio.hdfs.datanode.memory.max-mb	4096	The maximum allocation for the HdfsDataNode services, in MBs. The minimum defaults to 1024MB.
attivio.hdfs.journal.memory.max-mb	2048	The maximum allocation for the HdfsJournalNode services, in MBs. The minimum defaults to 1024MB.
attivio.hdfs.failover.controller.memory.max-mb	256	The maximum allocation for the HdfsFailoverController services, in MBs. The minimum defaults to 256MB.
dfs.replication	2	The number of replicas to maintain of each file in HDFS. A value greater than 1 is require for high availability (in the event the HdfsDataNode housing the file is unavailable.)

* the directory 'data-agent' is generated when the agent starts up, see AIE Agent page for additional information

HBase

Specify the nodes where the HBaseMaster and HBaseRegionServer services should run. In this example, nodes 6, 7 and 8 will be the Hadoop Master nodes and nodes 9 - 13 will be Hadoop Slave nodes.

```
    <hbase>
      <master host="host6"/>
      <master host="host7"/>
      <master host="host8"/>
      <regionserver host="host9"/>
      <regionserver host="host10"/>
      <regionserver host="host11"/>
      <regionserver host="host12"/>
      <regionserver host="host13"/>
      <properties>
        <property>
          <name>attivio.hbase.region.memory.max-mb</name>
          <value>5128</value>
        </property>
      </properties>
    </hbase>
```

The following properties can be configured for HBase, overriding their defaults.

Property	Default	Description
attivio.hbase.region.memory.max-mb	*3072*	The maximum allocation for the HBaseRegionServer services, in MBs. The minimum defaults to 1024MB.
attivio.hbase.master.memory.max-mb	1024	The maximum allocation for the HBaseMaster services, in MBs. The minimum defaults to 1024MB.

YARN

Specify the nodes where the YarnNodeManager, YarnResourceManager and JobHistoryService services should run. In this example, nodes 6, 7 and 8 will be the Hadoop Master nodes and nodes 9 - 13 will be Hadoop Slave nodes.

```
     <yarn>
      <nodemanager host="host9"/>
      <nodemanager host="host10"/>
      <nodemanager host="host11"/>
      <nodemanager host="host12"/>
      <nodemanager host="host13"/>
      <resourcemanager host="host6"/>
      <resourcemanager host="host7"/>
      <jobhistoryserver host="host7"/>
      <properties>
        <property>
          <name>yarn.scheduler.maximum-allocation-mb</name>
          <value>5125</value>
        </property>
        <property>
          <name>yarn.nodemanager.resource.memory-mb</name>
          <value>12288</value>
        </property>
      </properties>
    </yarn>
```

The following properties can be configured for YARN, overriding their defaults.

Property	Default	Description
yarn.scheduler.maximum-allocation-mb	4096	The maximum allocation for every container request at the RM, in MBs. Memory requests higher than this won't take effect, and will get capped to this value. This value should typically be increased.
yarn.scheduler.minimum-allocation-mb	512	The minimum allocation for each container request at the RM in MBs. Note that this value also serves as a granularity value, e.g., a container needing 513 MB will be rounded up to 2 x 512=1024 MB.
attivio.yarn.rm.memory.max-mb	1024	The maximum allocation for the YarnResourceManager services, in MBs. The minimum defaults to 1024MB.
attivio.yarn.nm.memory.max-mb	1024	The maximum allocation for the YarnNodeManager services, in MBs. The minimum defaults to 1024MB.
yarn.nodemanager.pmem-check-enabled	false	Do not change. Enable pmem checking. Attivio index writers/searchers run within YARN containers are Java processes and Java processes typically exceed their configured heap size leading up to a garbage collection event. If this property were set to **`true`** , those processes may be terminated by the YARN node manager, leading to instability.

Sample Topologies

The topology and server resources specified in this tutorial are for demonstration purposes only and do not represent an endorsed configuration for production purposes. A proper sizing exercise should be conducted to determine the required resources and topology for your project. If you would like assistance in completing such a sizing exercise, please contact Attivio and we'd be happy to assist.

Option 1 - External Cluster

```
<attivio xmlns="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.attivio.com/configuration/config classpath:/xsd/config.xsd">
  <topology agentPort="16999" projectdir=".">
    <zookeepers>
      <zookeeper host="externalzookeeper1" baseport="2181"/>
      <zookeeper host="externalzookeeper2" baseport="2181"/>
      <zookeeper host="externalzookeeper3" baseport="2181"/>
    </zookeepers>
    <hdfs/>
    <hbase/>
    <yarn/>
    <perfmonserver host="host1" baseport="16960"/>
    <nodes>
      <node name="node1" host="host1" baseport="17000"/>
      <node name="node2" host="host2" baseport="17000"/>
    </nodes>
    <nodesets>
      <nodeset name="admin">
        <include name="node1"/>
      </nodeset>
      <nodeset name="connectors">
        <include name="node1"/>
      </nodeset>
      <nodeset name="searchers">
        <include name="node1"/>
        <include name="node2"/>
      </nodeset>
      <nodeset name="ingestion">
        <include name="node1"/>
        <jvmArgument value="-XX:MaxGCPauseMillis=30000"/>
      </nodeset>
    </nodesets>
  </topology>
</attivio>
```

Option 2 - Attivio Cluster (Minimal)

```
<attivio xmlns="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.attivio.com/configuration/config classpath:/xsd/config.xsd">
  <topology agentPort="16999" projectdir=".">
    <zookeepers>
      <zookeeper host="host1" baseport="16980"/>
      <zookeeper host="host2" baseport="16980"/>
      <zookeeper host="host3" baseport="16980"/>
      <properties>
        <property>
          <name>dataLogDir</name> <!-- optional: specify a faster disk for ZooKepper write-ahead logs -->
          <value>/mnt/fastdisk/zoo</value>
        </property>
      </properties>
    </zookeepers>
    <hdfs>
      <namenode host="host1"/>
      <namenode host="host2"/>
      <failovercontrollernode host="host1"/>
      <failovercontrollernode host="host2"/>
      <journalnode host="host1"/>
      <journalnode host="host2"/>
      <journalnode host="host3"/>
      <datanode host="host1"/>
      <datanode host="host2"/>
      <datanode host="host3"/>
      <properties>
        <property>
```

```
          <name>attivio.hadoop.data.dir</name> <!-- optional -->
          <value>/opt/attivio/aiehadoopdata/</value>
        </property>
      </properties>
    </hdfs>
    <yarn>
      <nodemanager host="host1"/>
      <nodemanager host="host2"/>
      <nodemanager host="host3"/>
      <resourcemanager host="host1"/>
      <resourcemanager host="host2"/>
      <resourcemanager host="host3"/>
      <jobhistoryserver host="host3"/>
      <properties>
        <property>
          <name>yarn.scheduler.maximum-allocation-mb</name> <!-- maximum memory to allocate to any one
container -->
          <value>7168</value>
        </property>
        <property>
          <name>yarn.nodemanager.resource.memory-mb</name> <!-- total memory on nodemanager node for all
containers and Sliders -->
          <value>10240</value>
        </property>
      </properties>
    </yarn>
    <hbase>
      <master host="host1"/>
      <master host="host2"/>
      <regionserver host="host1"/>
      <regionserver host="host2"/>
      <regionserver host="host3"/>
    </hbase>
    <perfmonserver host="host1" baseport="16960"/>
    <nodes>
      <node name="one" host="host1" baseport="17000"/>
      <node name="two" host="host2" baseport="17000"/>
      <node name="three" host="host3" baseport="17000"/>
    </nodes>
    <nodesets>
      <nodeset name="connectors">
        <include name="three"/>
      </nodeset>
      <nodeset name="admin">
        <include name="one"/>
      </nodeset>
      <nodeset name="ingestion">
        <include name="three"/>
      </nodeset>
      <nodeset name="searchers">
        <include name="one"/>
        <include name="two"/>
      </nodeset>
    </nodesets>
  </topology>
</attivio>
```

Option 2: Attivio Cluster (Extended)

```
<attivio xmlns="http://www.attivio.com/configuration/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.attivio.com/configuration/config classpath:/xsd/config.xsd">
  <topology agentPort="16999" projectdir=".">
    <zookeepers>
      <zookeeper host="host6" baseport="16980"/>
      <zookeeper host="host7" baseport="16980"/>
      <zookeeper host="host8" baseport="16980"/>
    </zookeepers>
    <hdfs>
      <namenode host="host6"/>
      <namenode host="host7"/>
      <failovercontrollernode host="host6"/>
      <failovercontrollernode host="host7"/>
      <journalnode host="host6"/>
      <journalnode host="host7"/>
      <journalnode host="host8"/>
      <datanode host="host9"/>
      <datanode host="host10"/>
      <datanode host="host11"/>
      <datanode host="host12"/>
      <datanode host="host13"/>
      <datanode host="host14"/>
      <datanode host="host15"/>
      <datanode host="host16"/>
      <properties>
        <property>
          <name>attivio.hadoop.data.dir</name>
          <value>/opt/attivio/aiehadoopdata/</value>
        </property>
      </properties>
    </hdfs>
    <yarn>
      <nodemanager host="host9"/>
      <nodemanager host="host10"/>
      <nodemanager host="host11"/>
      <nodemanager host="host12"/>
      <nodemanager host="host13"/>
      <nodemanager host="host14"/>
      <nodemanager host="host15"/>
      <nodemanager host="host16"/>
      <resourcemanager host="host6"/>
      <resourcemanager host="host7"/>
      <jobhistoryserver host="host7"/>
      <properties>
        <property>
          <name>yarn.scheduler.maximum-allocation-mb</name>
          <value>5125</value>
        </property>
        <property>
          <name>yarn.nodemanager.resource.memory-mb</name>
          <value>12288</value>
        </property>
      </properties>
    </yarn>
    <hbase>
      <master host="host6"/>
      <master host="host7"/>
      <regionserver host="host9"/>
      <regionserver host="host10"/>
      <regionserver host="host11"/>
      <regionserver host="host12"/>
      <regionserver host="host13"/>
      <regionserver host="host14"/>
      <regionserver host="host15"/>
      <regionserver host="host16"/>
    </hbase>
    <perfmonserver host="host2" baseport="16960"/>
    <nodes>
      <node name="adminconn" host="host1" baseport="17000"/>
      <node name="ingest1" host="host2" baseport="17000"/>
      <node name="ingest2" host="host3" baseport="17000"/>
      <node name="query1" host="host4" baseport="17000"/>
      <node name="query2" host="host5" baseport="17000"/>
    </nodes>
    <nodesets>
      <nodeset name="connectors">
        <include name="adminconn"/>
        <include name="ingest2"/>
      </nodeset>
```

```
      <nodeset name="admin">
        <include name="adminconn"/>
      </nodeset>
      <nodeset name="ingestion">
        <include name="ingest1"/>
        <include name="ingest2"/>
      </nodeset>
      <nodeset name="searchers">
        <include name="query1"/>
        <include name="query2"/>
      </nodeset>
    </nodesets>
  </topology>
</attivio>
```

Modifications to Index.index.xml

Making Attivio work with multiple partitions requires changes in Attivio feature-configuration files. These files are found in the `<project-dir>\conf\features\core\` directory.

See Configure the Index for more information on configuring partitioning, index writers and searchers.

Modifications to BusinessCenter.xml

As Index.index.xml configures the primary index, the `<project-dir>\conf\features\businesscenter\BusinessCenter.xml` file configures the *abc-index* for the Business Center. Version 5.6.1 adds the following XML elements and attributes, performing the same functions as described in Configure the Index. Note that this is a subset of the functionality supported in the main index.

- memory@size and memory@diskCacheSize
- partitionSet@size
- writer@search
- searchers@rows

In most cases, the only consideration is to add `searchers` with `@rows` to 1 (or greater) for HA (or additional QPS capacity). In very rare cases, you may need more than one partition and to increase the memory.

Index Writer/Searcher Considerations and Cluster Feasibility

Keep the following rules in mind:

1. Each index writer or searcher (an *indexer component*) requires a single container in YARN.
2. Containers are allocated to nodes running a YarnNodeManager.
3. The number of containers required by an index is the number of columns multiplied by the number of rows.
4. An allocation is permissible on a given YARN nide when the node has enough unallocated memory to accommodate the container. (Under Option 1, additional constraints on CPU resources may apply if so configured in the external cluster environment.)
5. Additionally, each index requires a YARN application supported by the Slider application master. The application master also requires a container. If an index has under 16 components, the application master requires 512MB. Otherwise, it requires 1024MB.
6. Unless the default placement policy is changed, no two components of the same index is allowed to execute on the same node.
7. Typically, an Attivio deployment has two indices: the primary *index* and the Business Center *abc-index*. Omitting the Business Center module will omit the *abc-index*.
8. For high availability (HA), at the writers must have writer@search set to **`true`** and there must be at least one extra row of searchers. This applies to the as *abc-index* well.

Typically, you will have the same number of YARN nodes as main index colums x rows (or plus one extra to take over in the event of a single host failure). In such scenarios, you should ensure that the yarn.nodemanager.resource.memory-mb available is greater than or equal to the memory required by a component from each index plus the memory required by the application master for each index. This ensures that there is always enough memory to fully deploy every index. This cluster feasibility check is performed by the aie-cli. A warning will be displayed if the check fails or if the configuration is more complicated and a feasibility verification cannot be computed.

Layout

A layout defines how workflows and services are distributed across the nodes and nodesets in an Attivio deployment. The layout is comprised of `service-location` and `workflow-location` elements.

Default Nodesets

Default nodesets are required for services, for ingest, and for query workflows. These default locations are used for all workflows and services that are not explicitly mapped to other nodes or nodesets *or* implicitly mapped via a subflow reference (see below). The defaults are configured as follows:

```
    <layout default-service-nodeset="*" default-ingest-nodeset="ingestion" default-query-nodeset="searchers"
default-connector-nodeset="connectors">
      <service-location name="scheduler" nodeset="admin"/>
    </layout>
```

If unspecified, the `default-connector-nodeset` is set to the value of the `default-service-nodeset`.

Service, Connector and Workflow Locations

Service, connector, and workflow locations take a service/connector/workflow `name` attribute and a `nodeset` attribute. At start-up, each node uses these lists to determine whether to start the individual services/connectors/workflows. Generally any workflow without an explicit workflow location (as a subflow of another workflow) runs on the same nodeset as the referencing workflow. If multiple referencing workflows are running in different locations, the referenced workflow runs in the same location as the last referencing workflow.

You can configure a sample service location as follows:

```
<service-location name="webapp" nodeset="node1" />
```

You can configure a sample connector location as follows:

```
<connector-location name="myConnector" nodeset="connectorNodes" />
```

You can configure a sample workflow location as follows:

```
<workflow-location name="myIngestWf" nodeset="ingestion" scheme="loadBalanced" />
```

If the **`languagemodel`** project is included in your project, the languageModel feature will be added to your project configuration:

```
<lm:languageModel/>
```

Note: `lm` prefix represents new namespace: `http://www.attivio.com/configuration/features/languagemodel`

This automatically adds the language model service to all nodes in the cluster. To restrict the nodes running the language model service, specify the nodeset attribute:

```
<lm:languageModel nodeset="ingest"/>
```

To specify a different named language model service use the service attribute:

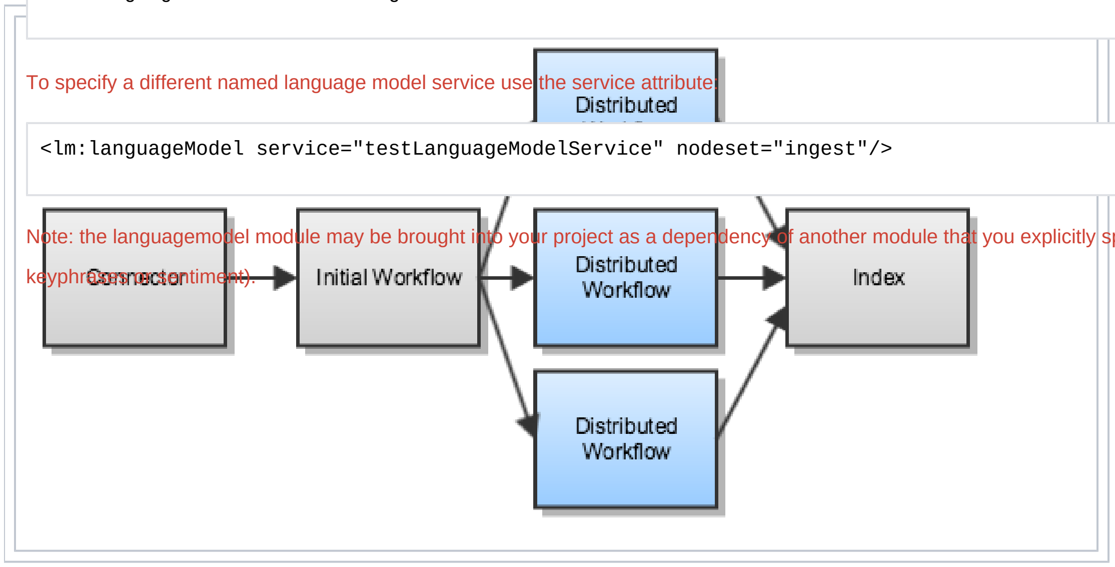
```
<lm:languageModel service="testLanguageModelService" nodeset="ingest"/>
```

Note: the languagemodel module may be brought into your project as a dependency of another module that you explicitly specify (for example keyphrases or sentiment).

Distribution Schemes

A distribution scheme determines how documents are distributed to the nodes in the farm of processing machines. The `workflow-location` descriptor has a `distribution scheme` attribute that determines how to distribute messages to workflows when referenced from another workflow.

For example, you might want a CPU-intensive workflow to run in parallel across a farm of machines. You could configure the system so that a single connector feeding a workflow does some preliminary processing. Then you could call a subflow that is distributed, before it rejoins the original workflow that is going to the indexer.



Attivio provides two different distribution schemes out of the box.

Scheme Name	Module	Description
loadbalanced	core-app	HTTP based scheme that performs basic load balancing across a series of nodes in a destination workflow's nodeset. Handles nodes of varying capacity and failed nodes gracefully.
defaultDistributionScheme	core-app	By default this is an instance of the *loadBalanced* scheme. If a workflow distribution scheme is not specified in a workflow-location, this is used.

Although this technique is useful for increasing the throughput of computationally intensive content processing, eventually, content indexing becomes a bottleneck, and additional content processing nodes have little effect on throughput.

When a workflow references a subflow running on the same nodeset, the distribution scheme is ignored and direct in-memory routing to the subflow is used.

Using the Agent and CLI

AIE-Agent is an independent executable that runs on every Attivio node. AIE-CLI is its Command-Line Interface. Using these tools we can start and stop the Configuration Servers, the Store (for single-node, unclustered topologies only), Performance Monitor service, and the Attivio nodes, even though they are located on multiple servers. For Option 2, Attivio Cluster topologies, we can also start and start the Hadoop services. We can also perform some simple monitoring of these processes to be sure they are all alive and well. Main article: Starting and Stopping Attivio.

Starting AIE-Agent

You'll need to start the AIE-Agent on each host separately.

To start the server on a specific host, navigate to **`<install-dir>\bin`** on that host and issue the following command:

```
<install-dir>\bin>aie-agent.exe -p 16999
```

Note that Agent can run as a service and might already be installed and running on your nodes.

Agent announces that it is running almost immediately:

```
i server on port 16999, poolSize=256
2014-06-11 09:06:33,210 INFO  Server - jetty-8.1.14.v20131031
2014-06-11 09:06:33,413 INFO  AbstractConnector - Started SelectChannelConnector
@0.0.0.0:16999
2014-06-11 09:06:33,413 INFO  Agent - Ready
```

Agent's default port is 16999.

Leave the agents running in the command windows. To stop them later, use **`control-C`** .

Starting AIE-CLI

AIE-CLI is the command-line interface to the Agent. Run it on any node where Attivio is installed and Agent is running.

```
<install-dir>\bin>aie-cli -p c:\attivio-projects\Factbook
```

AIE-CLI needs a path to the project files (**`-p`**). One AIE-CLI instance runs one AIE project.

Running the Project

AIE-CLI runs as a console. To start up the system, type **`start all`** at the AIE-CLI prompt.

With a single-node, unclustered topology, the Agent first starts the Configuration Server and deploys the project configuration files. Then it starts the Store. Next, the Performance Monitoring service is started. Then the Agent starts the AIE node. AIE-CLI gives you a status summary table at the bottom of the window.

```
aie> start all
Starting aie-configserver  at localhost:16980
Starting aie-store  at localhost:16970
uploading /conf/properties/core-app/attivio.core-app.properties
uploading /conf/properties/advancedtextextraction/advancedtextextraction.properies
uploading /conf/properties/factbook/factbook.properties
uploading /conf/properties/memory/memory.properties
4 resources uploaded
resource transfer to host localhost complete (ph=ProcessHandle [handle=3, pid=9
00, agentId=2ce444c6-4f99-447a-b6eb-28ece390545f])
resource transfer complete
Deploying project factbook to configuration server
Starting aie-perfmon  at node1:16960
Starting aie-node localhost at host1:17000
aie>



Updating: 1/1
     Nodes: UP(1) DOWN(0)                Stores: UP(1) DOWN(0)
CfgServers: UP(1) DOWN(0)               Perfmon: UP(1) DOWN(0)
```

With **Option 1 - External Cluster**, the Agent first connects to the ZooKeepers listed in `topology.xml` and deploys the project configuration files along with everything necessary to run Attivio in the external cluster. Then it requests the index processes be started in YARN. Next, the Performance Monitoring service is started, so it can monitor the individual nodes as they start. Then the Agent starts the AIE nodes in parallel. ZooKeeper copies the configuration files to each node as it begins to run. AIE-CLI gives you a status summary table at the bottom of the window.

```
aie> start all
Starting aie-perfmon  at node2:16960
Starting aie-node ingest1 at node2:17000
Starting aie-node adminconn at node1:17000
Starting aie-node ingest2 at node3:17000
Starting aie-node query2 at node5:17000
Starting aie-node query1 at node4:17000
Starting index: abc-index
  Please wait...
  Done.
Starting index: index
  Please wait...
  Done.
```

With **Option 2 - Attivio Cluster**, the Agent first starts the ZooKeepers listed in `topology.xml` and deploys the project configuration files along with everything necessary to run Attivio in the Hadoop cluster. Then the Hadoop processes are started in the proper order. Then it requests the index processes be started in YARN. Next, the Performance Monitoring service is started, so it can monitor the individual nodes as they start. Then the Agent starts the AIE nodes in parallel. ZooKeeper copies the configuration files to each node as it begins to run. AIE-CLI gives you a status summary table at the bottom of the window.

```
aie> start all
Starting the Hadoop cluster and waiting until all the services are running...
...The Hadoop cluster is running.
Starting aie-perfmon  at node2:16960
Starting aie-node ingest1 at node2:17000
Starting aie-node adminconn at node1:17000
Starting aie-node ingest2 at node3:17000
Starting aie-node query2 at node5:17000
Starting aie-node query1 at node4:17000
Starting index: abc-index
  Please wait...
  Done.
Starting index: index
  Please wait...
  Done.
```

See also...

Changes in the project's configuration files are typically needed. Remember that the Attivio nodes get their configuration files from the Configuration Server(s), not from the local file system. Your edits do not take effect until you upload the altered files to the configuration server.

Use the AIE-CLI's **deploy** command to re-deploy a project after you edit its configuration. The Agent will stop the individual Attivio nodes, deploy the new files to the Configuration Server(s), and then restart the nodes.

Monitoring the Project

The AIE-CLI lets us perform some simple monitoring. Type **status** at the prompt.

Single-node, unclustered:

```
aie> status

TYPE    NAME  HOST      BASEPORT HANDLE PID   STATUS
------------------------------------------------------------------------
config  null  localhost     16980    1      12056 RUNNING
perfmon null  localhost     16960    4      6484  RUNNING
store   null  localhost     16970    2      13672 RUNNING
node    local localhost     17000    6      9460  RUNNING
```

Option 1 - External Cluster:

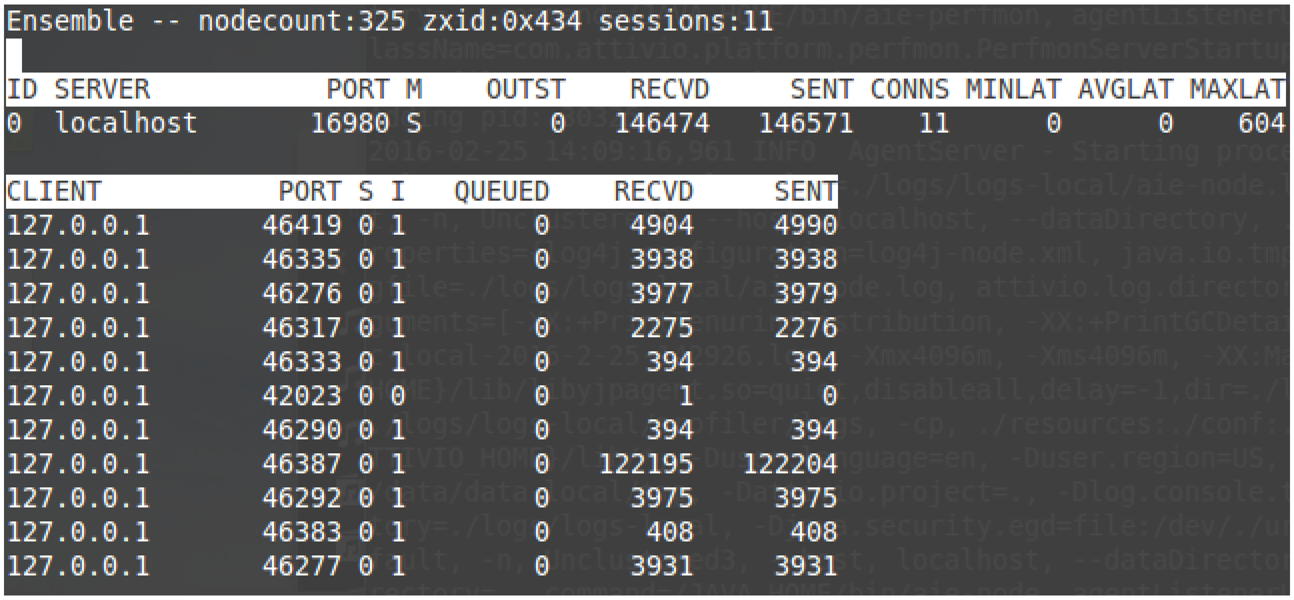
```
aie> status

TYPE    NAME      HOST        BASEPORT HANDLE PID   STATUS
------------------------------------------------------------------------
perfmon           node2 16960    4      11974 RUNNING
node    adminconn node1 17000    3      22415 RUNNING
node    ingest1   node2 17000    5      12200 RUNNING
node    ingest2   node3 17000    3      10734 RUNNING
node    query1    node4 17000    3      9305  RUNNING
node    query2    node5 17000    3      9246  RUNNING


abc-index [numPartitions=1, numRows=0, loadFactor=1]
          writer
--------------------
part0     ONLINE

 reader: 0 desired, 0 live, 0 active requests, 0 failed recently
 writer: 1 desired, 1 live, 0 active requests, 0 failed recently


index [numPartitions=1, numRows=0, loadFactor=1]
          writer
--------------------
part0     ONLINE

 reader: 0 desired, 0 live, 0 active requests, 0 failed recently
 writer: 1 desired, 1 live, 0 active requests, 0 failed recently
```

Option 2 - Attivio Cluster:

```
aie> status
Attivio Cluster (default)

TYPE    NAME      HOST        BASEPORT HANDLE PID   STATUS
------------------------------------------------------------------------
perfmon           node2 16960    4      11974 RUNNING
node    adminconn node1 17000    3      22415 RUNNING
node    ingest1   node2 17000    5      12200 RUNNING
node    ingest2   node3 17000    3      10734 RUNNING
node    query1    node4 17000    3      9305  RUNNING
node    query2    node5 17000    3      9246  RUNNING


abc-index [numPartitions=1, numRows=0, loadFactor=1]
          writer
--------------------
part0     ONLINE

 reader: 0 desired, 0 live, 0 active requests, 0 failed recently
 writer: 1 desired, 1 live, 0 active requests, 0 failed recently


index [numPartitions=1, numRows=0, loadFactor=1]
          writer
--------------------
part0     ONLINE

 reader: 0 desired, 0 live, 0 active requests, 0 failed recently
 writer: 1 desired, 1 live, 0 active requests, 0 failed recently


Hadoop Cluster Services
HOST         TYPE                   PID   STATUS
----------------------------------------------
node10  HBaseRegionServer      12074 RUNNING
node10  HdfsDataNode           11937 RUNNING
node10  YarnNodeManager        12159 RUNNING
node11  HBaseRegionServer      7738  RUNNING
node11  HdfsDataNode           7601  RUNNING
node11  YarnNodeManager        7819  RUNNING
node6  HBaseMaster            8182  RUNNING
node6  HdfsFailoverController 8027  RUNNING
node6  HdfsJournalNode        7822  RUNNING
node6  HdfsNameNode           7926  RUNNING
node6  YarnResourceManager    8185  RUNNING
node7  HBaseMaster            9118  RUNNING
node7  HdfsFailoverController 9012  RUNNING
node7  HdfsJournalNode        8777  RUNNING
node7  HdfsNameNode           8863  RUNNING
node7  YarnResourceManager    9121  RUNNING
node7  JobHistoryService    9542  RUNNING
node8  HdfsJournalNode        32633 RUNNING
node9  HBaseRegionServer      5607  RUNNING
node9  HdfsDataNode           5470  RUNNING
node9  YarnNodeManager        5691  RUNNING
```

Monitoring the ConfigServer

The `zkTop.py` lets us perform some simple monitoring of the config node which uses ZooKeeper. Type **bin/zkTop.py --server localhost:16980** at the prompt. Localhost is the HOST of the config and `16980` is the BASEPORT. This will show any latency issues with ZooKeeper.



```
Ensemble -- nodecount:325 zxid:0x434 sessions:11

ID SERVER          PORT M    OUTST    RECVD     SENT CONNS MINLAT AVGLAT MAXLAT
0  localhost      16980 S        0   146474   146571    11      0      0    604

CLIENT           PORT S I   QUEUED    RECVD     SENT
127.0.0.1       46419 0 1        0     4904     4990
127.0.0.1       46335 0 1        0     3938     3938
127.0.0.1       46276 0 1        0     3977     3979
127.0.0.1       46317 0 1        0     2275     2276
127.0.0.1       46333 0 1        0      394      394
127.0.0.1       42023 0 0        0        1        0
127.0.0.1       46290 0 1        0      394      394
127.0.0.1       46387 0 1        0   122195   122204
127.0.0.1       46292 0 1        0     3975     3975
127.0.0.1       46383 0 1        0      408      408
127.0.0.1       46277 0 1        0     3931     3931
```

Checking Stderr and Stdout Files

Note that the "handle" number of a process in the **`status`** display keys to a directory that holds the output files for that process. The files will be found in this location:

```
<install-dir>\bin\data-agent\procinfo\<handle-number>\
```

The **`<handle-number>`** directory contains the **`command.txt`, hadoop.txt, `stderr.txt`, `stdout.txt`,** and **`workingDirectory.txt`** files for that process.

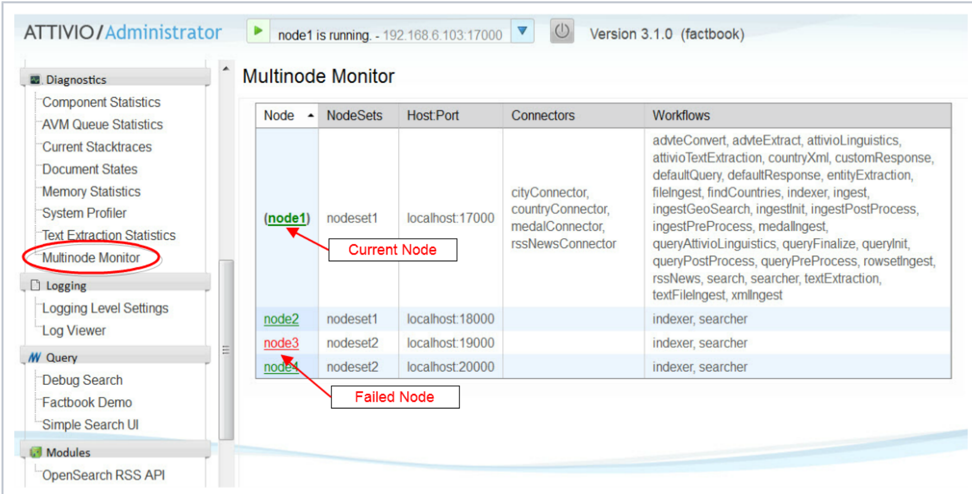
Stopping the Project

To shut down the project (across multiple servers), simple type **`stop all`** at the AIE-CLI prompt.

Multi-Node Administration

Each node has its own AIE AIE Administrator (Admin UI): `http://<hostname>:<bast port number>/admin`

You can administer all nodes from the Multi-Node Monitor in **Diagnostics** section if any node's administration page.



Multinode Monitor

Node	NodeSets	Host:Port	Connectors	Workflows
(node1)	nodeset1	localhost:17000	cityConnector, countryConnector, medalConnector, rssNewsConnector	advteConvert, advteExtract, attivioLinguistics, attivioTextExtraction, countryXml, customResponse, defaultQuery, defaultResponse, entityExtraction, fileIngest, findCountries, indexer, ingest, ingestGeoSearch, ingestInit, ingestPostProcess, ingestPreProcess, medalIngest, queryAttivioLinguistics, queryFinalize, queryInit, queryPostProcess, queryPreProcess, rowsetIngest, rssNews, search, searcher, textExtraction, textFileIngest, xmlIngest
node2	nodeset1	localhost:18000		indexer, searcher
node3	nodeset2	localhost:19000		indexer, searcher
node4	nodeset2	localhost:20000		indexer, searcher

The monitor provides a convenient overview of all nodes in the system.

Backup and Restore

See the Backup and Restore for Attivio Clusters page for a discussion of restoring the index to a previous version in the event of a multi-node failure.