



Installation and Configuration Guide

Version: 1.1.0
2/1/2017

Table of Contents

1. Introduction	3
1.1. Bagri Overview.....	3
2. Requirements to Install Bagri.....	4
2.1. Hardware Requirements	4
2.2. Software Requirements.....	4
2.3. Deployment Options.....	5
3. Bagri Installation Package.....	7
4. Initial Configuration to run the Product	9
5. Run Bagri.....	11
5.1. Run the Product on Windows	11
5.2. Run the Product on Linux.....	12
5.3. Run the Administrative Server.....	14
6. Configure Graphical User Interface (optional)	17
7. Appendix A: Configuration Files.....	21
7.1. Configuration file access.xml	21
7.2. Configuration file config.xml.....	23
7.3. Configuration file admin.properties	37
7.4. Configuration files <profile_name>.properties	37
8. Related Documentation	39

1. Introduction

This guide is intended for administrators who are going to install and configure Bagri. The guide provides detailed instructions on how best to deploy and set up the product. It lists all product requirements, necessary rights and permissions and guides you through the installation and configuration processes.

After reading this guide you will be able to perform initial configuration for the product and run it based on the selected deployment option.

1.1. Bagri Overview

Bagri is a Document Database built on top of distributed cache solution like [Hazelcast](#), [Coherence](#), etc. The system allows to process semi-structured schema-less documents and perform distributed queries on them in real-time. System scales horizontally very well using data sharding technology, when all documents are distributed evenly between distributed cache partitions. Started as Native XML DB initially, the system has grown to process any kind of documents with self-describing data format (JSON, for instance) via pluggable parsers API.

The system is best suited to work with medium (1K..1M) sized documents containing data (data-centric XML documents, as opposite to content-centric XML). The number of simultaneously processed documents is limited by the size of internal distributed data store only. The underlying cache platform is used not only as a Data Grid, but also as a Computation Grid for parallel data processing by means of tasks performing on distributed cache cluster.

2. Requirements to Install Bagri

This section provides the requirements for the computer where Bagri is going to be installed. Refer to the following sections for detailed information:

- [Hardware Requirements](#)
- [Software Requirements](#)
- [Deployment Options](#)

2.1. Hardware Requirements

Before installing Bagri, make sure that your hardware meets the following requirements:

Hardware Component	Minimum	Recommended
Processor	No special requirements	Intel Corei3..i7, 2..4GHz
RAM	2 GB	6 – 8 GB per node, depending on the database size NOTE: Before running the Product, please make sure that some rough calculation of the memory required has been performed. The default value for 1 node is set to 4 GB, if you need to change the default configuration, you can manually specify the required value in the configuration file. Please refer to the Initial Configuration to run the Product section for more details.
Disk Space	50MB for initial installation	The disk space required for Bagri to function properly depends on the database size; the average size for each node can be from 10 to 100 GB or more.

2.2. Software Requirements

The table below lists the minimum software requirements for Bagri:

Component	Requirements
Operating system	<ul style="list-style-type: none">• Desktop (Client) OS: Microsoft Windows 7 and above SUSE Linux v.11 SP4

Component	Requirements
	<p>openSUSE Tumbleweed Any other *nix distribution with Java installed</p> <ul style="list-style-type: none">• Server OS: Microsoft Windows Server 2008 (64-bit) and above Microsoft Windows 7 (64-bit) and above SUSE Linux Enterprise Server 11 SP4 (64-bit) Red Hat Enterprise Linux 7 (64-bit) Any other 64-bit *nix distribution with Java installed
Additional Software	<ul style="list-style-type: none">• Java 1.7 or higher NOTE: Java is a mandatory requirement for the product. Please refer to Oracle documentation portal for instructions on how to install Java on Windows or Linux environment.• VisualVM 1.3.6 or higher NOTE: VisualVM or other JMX console (like JConsole, which is a part of JDK installation) can be used to monitor and manage system behavior.

2.3. Deployment Options

This section provides recommendations on how best to deploy Bagri product. Review these recommendations and choose the most suitable option depending on the amount of documents you are going to process with Bagri.

The system has a "schema" concept - the same as a schema in RDBMS. Each schema contains its own set of caches and components deployed in a distributed cache cluster. It allows the system to be deployed as following:

Option	Description
standalone Java application with embedded document database	<p>This option can be used for relatively small applications requiring to process some limited number of documents only. All documents are processed within one Java Virtual Machine and usually in one schema.</p> <p>Such deployment allows to achieve maximum performance.</p>
client-server application with distributed document database	<p>This option can be used for processing greater amounts of documents with several schemas applied. Clients can access schemas deployed in a separate distributed caches via</p>

Option	Description
	<p>provided XQJ_driver or via Bagri client (XDM) API. System scaling can be performed dynamically by adding new nodes to existing schema clusters which does not require system downtime.</p> <p>Such deployment is best suited for on-line processing of terabytes of data with memory-access speed.</p>

3. Bagri Installation Package

This chapter provides detailed overview of the installation packages for both Windows and Linux platforms.

The installation package is provided as an archive file:

- **bagri-1.1.0.zip** for Windows based machine;
- **bagri-1.1.0.tar.gz** for Linux based machine;

where digits represent release version of the product.

It consists of the following folders:

Folder name	Description
bin	<p>This folder contains the scripts that start Bagri Administrative and Data servers:</p> <ul style="list-style-type: none">• <code>bgadmin.cmd</code> – launches Administrative server on Windows platform• <code>bgadmin.conf</code> – supporting configuration file to run Administrative server on Linux platform• <code>bgadmin.sh</code> – launches Administrative server on Linux platform• <code>bgcache.cmd</code> – launches Data servers on Windows platform• <code>bgcache.conf</code> – supporting configuration file to run Data server on Linux platform• <code>bgcashe.sh</code> – launches Data server on Linux platform• <code>bgstop.cmd</code> – stops Data server(s) on Windows platform
config	<p>This folder contains sample Bagri profiles and their configuration settings. The folder consists of the following files:</p> <ul style="list-style-type: none">• <code>access.xml</code> – contains basic access settings.• <code>config.xml</code> – contains configuration settings for the profiles.• <code>first.properties</code>, <code>second.properties</code>, etc. – initial profile parameters.• <code>logging.xml</code>, <code>hz-logging.xml</code> – logging configurations. <p>Please refer to the Appendix A: Configuration Files section below for the detailed overview of all the configuration parameters.</p>
data	<p>This folder contains subfolders with sample documents that can be used for the system evaluation:</p> <ul style="list-style-type: none">• <code>json</code> – contains sample JSON documents• <code>tpox</code> – contains sample TPoX documents and queries• <code>xmark</code> – contains sample XMark document and queries
distr	<p>This folder contains redistributable client artifacts i.e. JSR-225 XQJ driver and Bagri plugin for VisualVM console</p>

Folder name	Description
docs	This folder contains Bagri documentation: <ul style="list-style-type: none">• javadocs explaining APIs and classes exposed via system libraries• this file providing installation and configuration instructions
lib	This folder contains Bagri server-side jars and their dependencies
samples	This folder contains sample programs which demo various system features

NOTE: The system supports this default folders structure. If you want to change the default setup, please make sure that bin, config and lib folders are located in the same directory.

4. Initial Configuration to run the Product

This section provides detailed procedures to be performed to start use the system and set up Bagri environment.

To Perform Initial Configuration

1. [Download](#) Bagri.

NOTE: Before performing the initial configuration, make sure that your user is granted all necessary permissions on the workstation where the product will run.

2. Extract the installation package.
3. Specify the required memory allocation for cache nodes used by the system. By default, the size of nodes is set to 4 GB. To change it, perform the following steps:

- For Windows based machine:

- Navigate to Bagri Installation directory → bin and open bgcache.cmd in a text editor.
- Find the memory allocation parameter and set it to the required value:

```
rem specify the JVM heap size  
set memory=4g
```

- Save the file.

For Linux based machine:

- Navigate to Bagri Installation directory → bin and open bgcache.conf in a text editor.
- Find the memory allocation parameters and set them to the required value:

```
main='com.bagri.server.hazelcast.BagriCacheServer'  
JAVA_OPTS=-Xms4g -Xmx4g
```

- Save the file.

4. Choose profile you want to launch. The system is delivered with four demo profiles located in the config directory (first.properties, second.properties, etc).

- Navigate to Bagri Installation directory → config and open <profile_name>.properties file in a text editor.
- Take schema name value from the choosen profile:

```
bdb.cluster.node.schemas=default
```

5. Specify connection settings for the profile to be used to connect to Administrative server. This step is optional and should be done in case you plan to use Administrative server to monitor the systems's behavior.

- Open the profile file and specify Administrative server host, for instance:

```
bdb.cluster.members=192.168.10.100
```

- Save the file

6. Specify network settings to be used by the chosen schema:

- Navigate to Bagri Installation directory → config and open config.xml in a text editor.
- Locate corresponding schema properties subsection and set schema ports range (bdb.schema.ports.first, bdb.schema.ports.last) and schema members list (bdb.schema.members) to the required values. You can find detailed information for all schema properties in the [Configuration file config.xml](#) section of this document.

```
<schema name="TPoX" active="true">
.....
<description>TPoX: schema for TPoX-related tests</description>
<properties>
  <entry name="bdb.schema.ports.first">10000</entry>
  <entry name="bdb.schema.ports.last">10100</entry>
  <entry name="bdb.schema.members">localhost</entry>
  <entry name="bdb.schema.thread.pool">16</entry>
  <entry name="bdb.schema.store.data.path">../data/tpox</entry>
  <entry name="bdb.schema.format.default">XML</entry>
  <entry name="bdb.schema.store.type">File</entry>
.....
</schema>
```

NOTE: Several member addresses can be specified as comma-separated list. If cluster nodes are located in the same sub-net, then range characters ('*' and '-') can be used. For instance, 10.3.10.* refers to IPs between 10.3.10.0 and 10.3.10.255. Address 10.3.10.4-18 refers to IPs between 10.3.10.4 and 10.3.10.18 (4 and 18 included). Domain names can be used as well.

- Save the file.

7. If you want to manage schema settings and monitor its behavior via GUI you can launch Administrative server. To connect Administrative server to already running and/or launched in the future Data servers:

- Navigate to Bagri Installation directory → config and open admin.properties file in a text editor.

- Specify hosts you'd like to connect to:

```
bdb.cluster.members=192.168.10.10-50
```

- Save the file.

5. Run Bagri

This chapter provides step-by-step instructions on how to run Bagri product. Refer to the following sections for detailed information:

- [Run the Product on Windows](#)
- [Run the Product on Linux](#)
- [Run the Administrative Server](#)

5.1. Run the Product on Windows

To Run Bagri

1. Navigate to **Start**→**Run** and type "cmd". Input change directory command and the path to the folder where bgcache.cmd is located and press **Enter**.

```
cd installation_folder\bagri-X.X.X\bin
```

2. Input the following command:

```
bgcache.cmd <profile-name> <node-number>
```

The command parameters are:

- bgcache.cmd: name of the command script to launch Bagri Data server
- profile-name: the name of profile file containing initial connectivity parameters to connect to cluster. This file is located in the config folder of the installation package. Several sample profiles are provided together with the product: first, second, third. If this parameter is not specified the application takes the default profile value equal to first.
- node-number: the number of Data server node instance when it is started as part of a cluster. If this parameter is not specified the application takes the default value equal to 0.

NOTE: Make sure that several Data servers are not started with the same node-number value on any machine where the system is running.

3. On the successful launch of the application, the system will log output to the console:

```

C:\Windows\system32\cmd.EXE - bgcacheload second 0
orderingMode=2, bdb.schema.trans.backup.async=0, bdb.schema.dict.backup.async=1, bdb.schema.trans.backup.read
=false, bdb.schema.store.type=File, bdb.schema.dict.backup.read=true, bdb.schema.thread.pool=16, bdb.schema.po
rts.last=10100, bdb.schema.data.backup.sync=0, bdb.schema.query.backup.read=true, bdb.schema.store.enabled=true
e, bdb.schema.population.size=1, xqj.schema.queryLanguageTypeAndVersion=1, bdb.schema.ports.first=10000, xqj.s
chema.bindingMode=0, bdb.schema.store.data.path=../data/tpox, bdb.schema.data.backup.read=false, bdb.schema.qu
ery.cache=true, bdb.schema.health.threshold.high=0, xqj.schema.boundarySpacePolicy=1, xqj.schema.scrollability
=1, bdb.schema.store.tx.buffer.size=2048, bdb.schema.partition.count=271, xqj.schema.holdability=2, bdb.schema
.query.parallel=true, bdb.schema.data.stats.enabled=true, bdb.schema.partition.pool=32, bdb.schema.password=IP
oX, bdb.schema.buffer.size=64, xqj.schema.copyNamespacesModePreserve=1, xqj.schema.queryTimeout=0, xqj.schema
.defaultFunctionNamespace=http://www.w3.org/2005/xpath-functions, xqj.schema.defaultElementTypeNamespace=http:/
/www.w3.org/2001/XMLSchema, bdb.schema.members=192.168.1.50, 192.168.1.139, bdb.schema.transaction.timeout=600
00, bdb.schema.health.threshold.low=25, xqj.schema.copyNamespacesModeInherit=1, bdb.schema.data.cache=NEVER, x
qj.schema.baseUri=file://../data/tpox/, xqj.schema.defaultOrderForEmptySequences=2, bdb.schema.format.default=X
ML, bdb.schema.query.backup.async=0, bdb.schema.population.buffer.size=1000000, xqj.schema.defaultCollationUri
=http://www.w3.org/2005/xpath-functions/collation/codepoint, xqj.schema.constructionMode=1, bdb.schema.data.ba
ckup.async=1, bdb.schema.trans.backup.sync=1, indexes=[Index {name=IDX_Customer_id, version=1, docType=<http
://tpox-benchmark.com/custacc>Customer, typePath=null, path=<http://tpox-benchmark.com/custacc>Customer/Eid,
dataType=<http://www.w3.org/2001/XMLSchema>string, caseSensitive=true, range=false, unique=true, created at=Mo
n Jan 25 13:36:26 MSK 2016, by=admin, description=Customer id, enabled=true}], Index {name=IDX_Security_Symbol,
version=1, docType=<http://tpox-benchmark.com/security>Security, typePath=null, path=<http://tpox-benchmark
.com/security>Security/<http://tpox-benchmark.com/security>Symbol/text(), dataType=<http://www.w3.org/2001/XML
Schema>string, caseSensitive=true, range=false, unique=true, created at=Wed May 25 13:36:26 MSK 2016, by=admin
, description=Security Symbol, enabled=true}], Index {name=IDX_Order_ID, version=1, docType=<http://www.fixpro
tocol.org/FIXML-4-4>FIXML, typePath=null, path=<http://www.fixprotocol.org/FIXML-4-4>FIXML/<http://www.fixpro
tocol.org/FIXML-4-4>Order/EID, dataType=<http://www.w3.org/2001/XMLSchema>string, caseSensitive=true, range=fa
lse, unique=true, created at=Wed May 25 13:36:26 MSK 2016, by=admin, description=Order ID, enabled=true}], tri
ggers=[], fragments=[], collections=[Collection {id=1, name=CLN_Security, version=1, docType=<http://tpox-ben
chmark.com/security>Security, created at=Fri May 20 01:01:26 MSK 2016, by=admin, description=All security docu
ments, enabled=true}], Collection {id=2, name=CLN_Customer, version=1, docType=<http://tpox-benchmark.com/cust
acc>Customer, created at=Fri May 20 01:01:26 MSK 2016, by=admin, description=All customer documents, enabled=t
rue}], Collection {id=3, name=CLN_Order, version=1, docType=<http://www.fixprotocol.org/FIXML-4-4>FIXML, creat
ed at=Fri May 20 01:01:26 MSK 2016, by=admin, description=All order documents, enabled=true}], Collection {id=4
, name=CLN_Custom, version=1, docType=null, created at=Fri May 20 01:01:26 MSK 2016, by=admin, description=Cus
tom collection, enabled=true}], resources=[Resource {name=tpox, version=0, path=tpox, description=TPoX resour
ce exposed via REST, created at=Tue Oct 04 13:36:26 MSK 2016, by=admin, module=rest_module, enabled=true}], in
itialized
2017-02-01 00:56:50.080 [hz.TPoX.cached.thread-23] INFO com.bagri.server.hazelcast.task.schema.SchemaPopulato
r - populateSchema; transactions size after loadAll: 0
2017-02-01 00:56:50.080 [hz.TPoX.partition-operation.thread-21] INFO com.hazelcast.map.impl.recordstore.DefaultRecordStore - [192.168.1.87]:10000 [TPoX] [3.7.5] Starting to load all keys for map xdm-document on partiti
onId=245
2017-02-01 00:56:50.185 [hz.TPoX.cached.thread-15] INFO com.bagri.server.hazelcast.store.FileDocumentCacheSto
re - ensureRepository; repo: com.bagri.server.hazelcast.impl.SchemaRepositoryImplP3d261033
2017-02-01 00:56:50.190 [hz.TPoX.cached.thread-15] INFO com.bagri.server.hazelcast.store.FileDocumentCacheSto
re - loadAllKeys.exit; returning keys: ?
2017-02-01 00:56:52.206 [hz.TPoX.cached.thread-23] INFO com.bagri.server.hazelcast.task.schema.SchemaPopulato
r - populateSchema; documents size after loadAll: ?
2017-02-01 00:56:52.221 [hz.TPoX.cached.thread-23] INFO com.bagri.server.hazelcast.task.schema.SchemaPopulato
r - call.exit; schema TPoX populated: true
  
```

Once the system is started, it writes all important events into the log files for every running server node. Every server node writes in its own log file that can be found in the directory Bagri Installation directory → logs → <profile_name> → cache.

5.2. Run the Product on Linux

To Run Bagri

1. To run Bagri on a Linux OS you can use any terminal software such as PuTTY. Change current directory to the bin folder where bgcacheload.sh is located and press **Enter**.

```
cd installation_folder\bagri-X.X.X\bin
```

2. Input the following command:

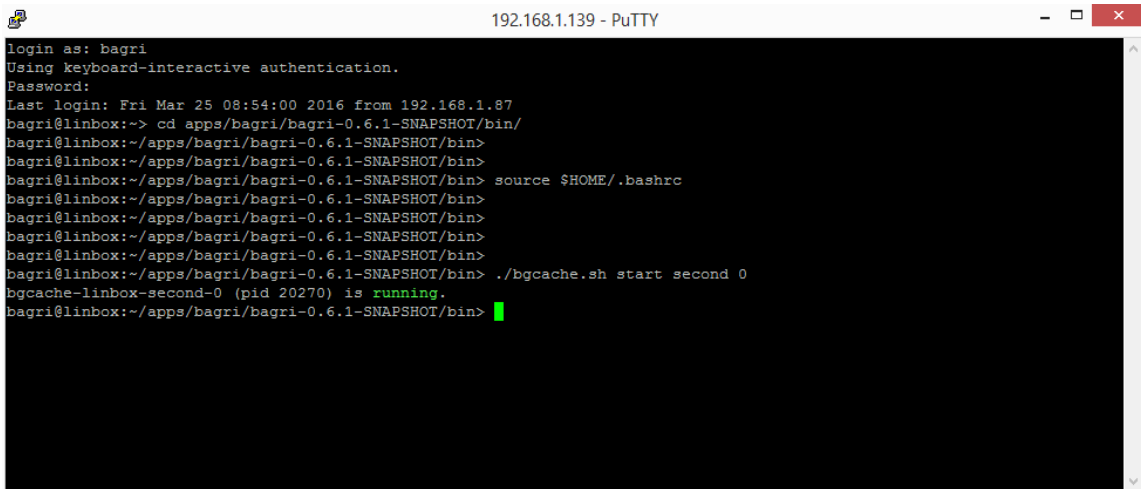
```
>./bgcacheload.sh start profile-name node-number
```

The command parameters are:

- bgcacheload.sh: name of the shell script to launch Bagri Data server.
- start: command to start new node instance
- profile-name: the name of profile file containing initial connectivity parameters to connect to cluster. This file is located in the config folder of the installation

package. If this parameter is not specified the application takes the default profile value equal to first.

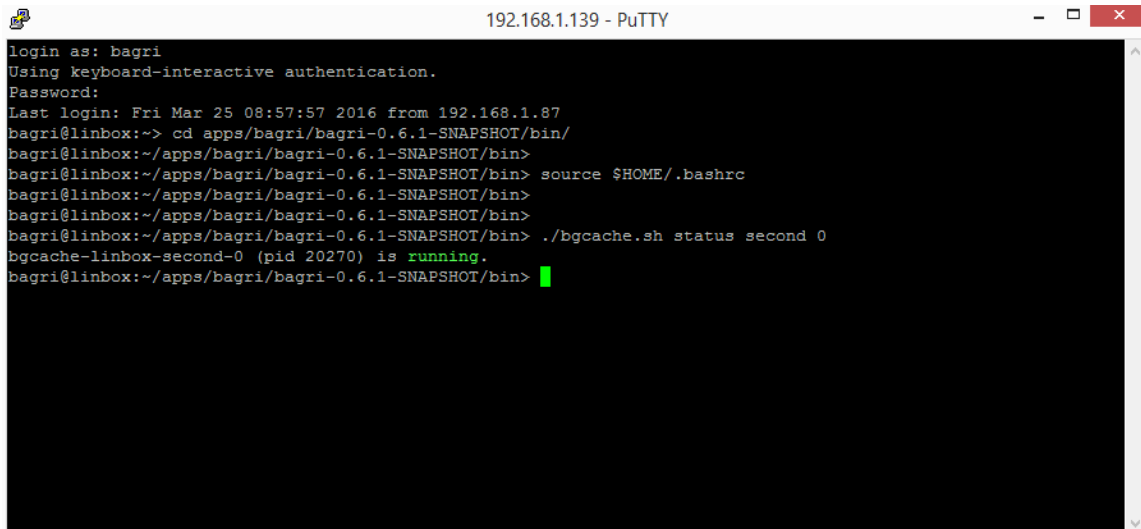
- node-number: the number of Data server node instance when it is started as part of a cluster. If this parameter is not specified the application takes the default value equal to 0.
3. On successful launch of the system, OS will display the message that the node has been started properly.



```
login as: bagri
Using keyboard-interactive authentication.
Password:
Last login: Fri Mar 25 08:54:00 2016 from 192.168.1.87
bagri@linbox:~> cd apps/bagri/bagri-0.6.1-SNAPSHOT/bin/
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> source $HOME/.bashrc
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> ./bgcache.sh start second 0
bgcache-linbox-second-0 (pid 20270) is running.
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
```

4. Once the system is started, it writes all important events into the log files for every running server node. Every server node writes in its own log file that can be found in the directory Bagri Installation directory → logs → <profile_name> → cache.
5. To check status of the node run the following command:

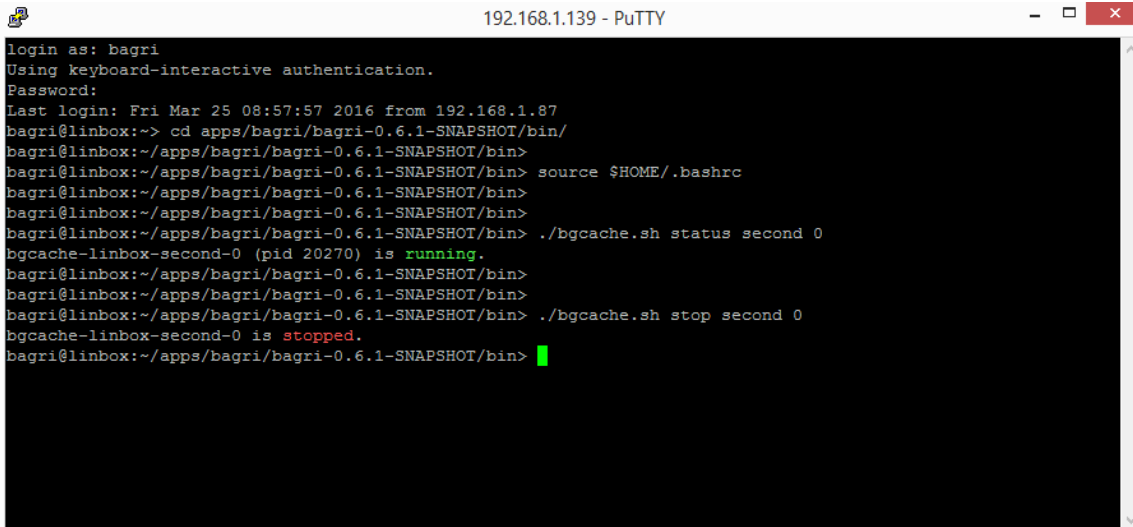
 >./bgcache.sh status profile-name node-number



```
login as: bagri
Using keyboard-interactive authentication.
Password:
Last login: Fri Mar 25 08:57:57 2016 from 192.168.1.87
bagri@linbox:~> cd apps/bagri/bagri-0.6.1-SNAPSHOT/bin/
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> source $HOME/.bashrc
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> ./bgcache.sh status second 0
bgcache-linbox-second-0 (pid 20270) is running.
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
```

6. To stop the node run the following command:

 >./bgcache.sh stop profile-name node-number



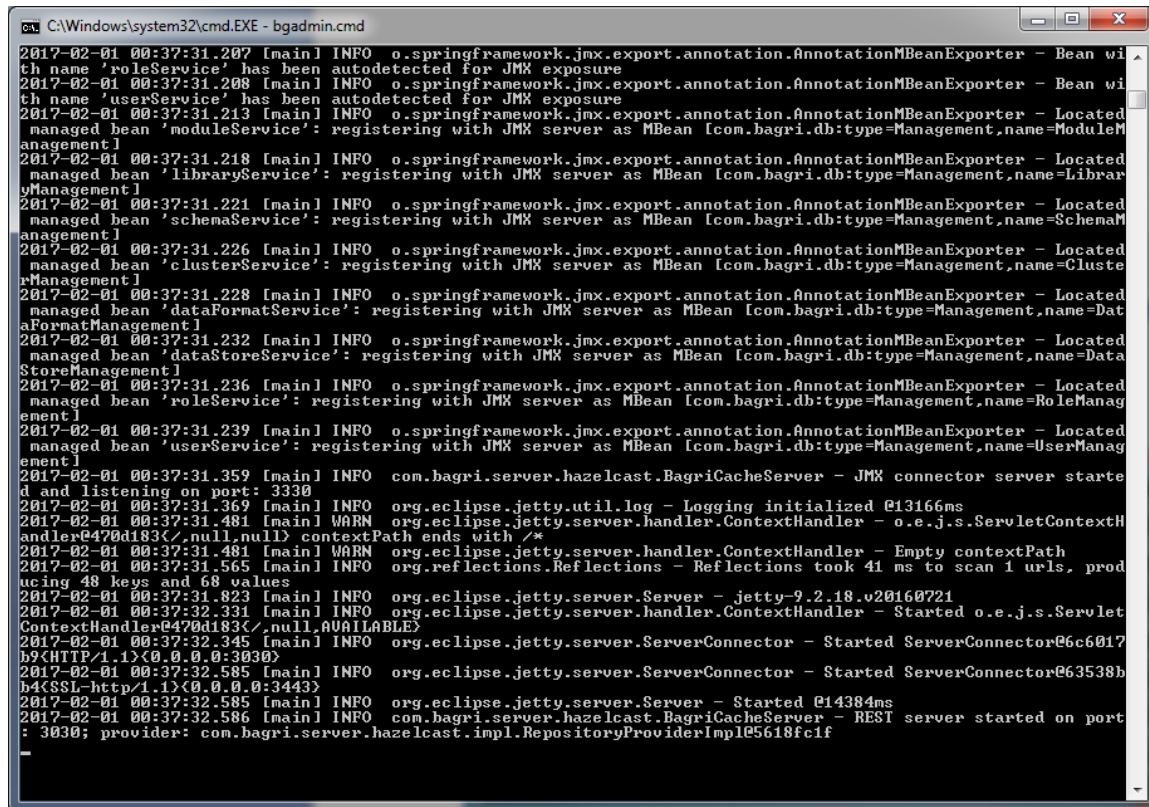
```
login as: bagri
Using keyboard-interactive authentication.
Password:
Last login: Fri Mar 25 08:57:57 2016 from 192.168.1.87
bagri@linbox:~> cd apps/bagri/bagri-0.6.1-SNAPSHOT/bin/
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> source $HOME/.bashrc
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> ./bgcache.sh status second 0
bgcache-linbox-second-0 (pid 20270) is running.
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> ./bgcache.sh stop second 0
bgcache-linbox-second-0 is stopped.
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> █
```

5.3. Run the Administrative Server

The Administrative server that is provided with the product allows centralized management of the system configuration as well as collection and monitoring of various system statistics. It allows to specify and deploy new profiles and/or change configuration of already deployed profiles. Please refer to the procedures below on how to run the Administrative server on Windows and Linux based machines.

To Run Administrative Server on Windows based machine

1. Navigate to **Start→Run** and type "cmd". Input change directory command and the path to the folder where bgadmin.cmd is located and press **Enter**.
`cd installation_folder\bagri-X.X.X\bin`
2. Input the following command:
`bgadmin.cmd`
3. On the successful launch of the application, the system will log output to the console:



```
C:\Windows\system32\cmd.EXE - bgadmin.cmd
2017-02-01 00:37:31.207 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Bean with name 'roleService' has been autodetected for JMX exposure
2017-02-01 00:37:31.208 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Bean with name 'userService' has been autodetected for JMX exposure
2017-02-01 00:37:31.213 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'moduleService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=ModuleManagement
2017-02-01 00:37:31.218 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'libraryService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=LibraryManagement
2017-02-01 00:37:31.221 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'schemaService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=SchemaManagement
2017-02-01 00:37:31.226 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'clusterService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=ClusterManagement
2017-02-01 00:37:31.228 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'dataFormatService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=DataFormatManagement
2017-02-01 00:37:31.232 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'dataStoreService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=DataStoreManagement
2017-02-01 00:37:31.236 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'roleService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=RoleManagement
2017-02-01 00:37:31.239 [main] INFO o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located managed bean 'userService': registering with JMX server as MBean Icom.bagri.db:type=Management,name=UserManagement
2017-02-01 00:37:31.359 [main] INFO com.bagri.server.hazelcast.BagriCacheServer - JMX connector server started and listening on port: 3330
2017-02-01 00:37:31.369 [main] INFO org.eclipse.jetty.util.log - Logging initialized @13166ms
2017-02-01 00:37:31.481 [main] WARN org.eclipse.jetty.server.handler.ContextHandler - o.e.j.s.ServletContextHandler@470d183c</,null,null> contextPath ends with /*
2017-02-01 00:37:31.481 [main] WARN org.eclipse.jetty.server.handler.ContextHandler - Empty contextPath
2017-02-01 00:37:31.565 [main] INFO org.reflections.Reflections - Reflections took 41 ms to scan 1 urls, producing 48 keys and 68 values
2017-02-01 00:37:31.823 [main] INFO org.eclipse.jetty.server.Server - jetty-9.2.18.v20160721
2017-02-01 00:37:32.331 [main] INFO org.eclipse.jetty.server.handler.ContextHandler - Started o.e.j.s.ServletContextHandler@470d183c</,null,AVAILABLE>
2017-02-01 00:37:32.345 [main] INFO org.eclipse.jetty.server.ServerConnector - Started ServerConnector@6c6017b9<HTTP/1.1><0.0.0.0:3030>
2017-02-01 00:37:32.585 [main] INFO org.eclipse.jetty.server.ServerConnector - Started ServerConnector@63538bb4<SSL-http/1.1><0.0.0.0:3443>
2017-02-01 00:37:32.585 [main] INFO org.eclipse.jetty.server.Server - Started @14384ms
2017-02-01 00:37:32.586 [main] INFO com.bagri.server.hazelcast.BagriCacheServer - REST server started on port : 3030; provider: com.bagri.server.hazelcast.impl.RepositoryProviderImpl@5618fc1f
```

4. To stop the admin node just close the window where the system was started or enter the following command:

Ctrl + C→Y

NOTE: Starting from version 1.1 Bagri ships with embedded REST server which is deployed on Administrative server by default. See the log message about the launched REST server as on the screenshot above.

To Run Administration Server on Linux based machine

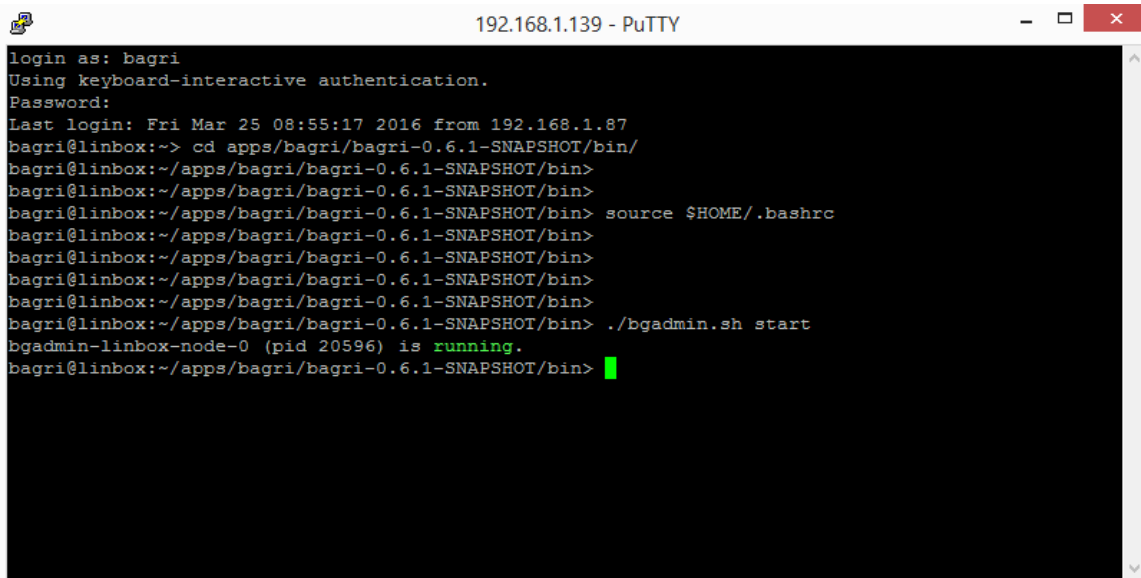
1. To run Administration server on a Linux OS you can use any terminal software such as PuTTY. Change directory to the bin folder where bgadmin.sh is located and press **Enter**.

```
cd installation_folder\bagri-X.X.X\bin
```

2. Input the following command:

```
>./bgadmin.sh start
```

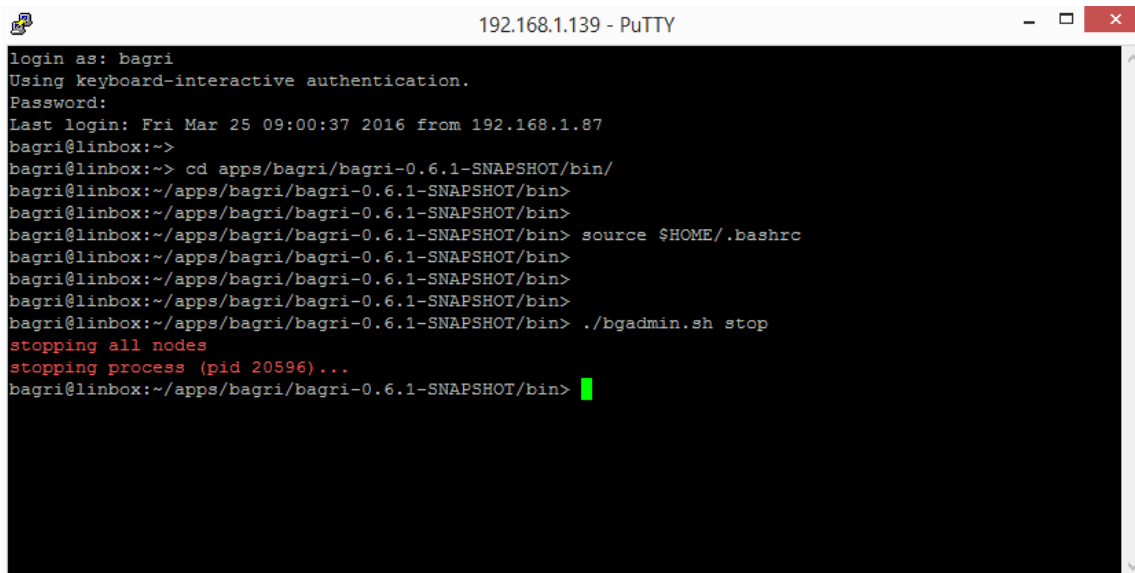
On successful launch of the server, OS will display the message that the node has been started properly:



```
192.168.1.139 - PuTTY
login as: bagri
Using keyboard-interactive authentication.
Password:
Last login: Fri Mar 25 08:55:17 2016 from 192.168.1.87
bagri@linbox:~> cd apps/bagri/bagri-0.6.1-SNAPSHOT/bin/
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> source $HOME/.bashrc
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> ./bgadmin.sh start
bgadmin-linbox-node-0 (pid 20596) is running.
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
```

3. To stop the admin server enter the following command:

>./bgadmin.sh stop



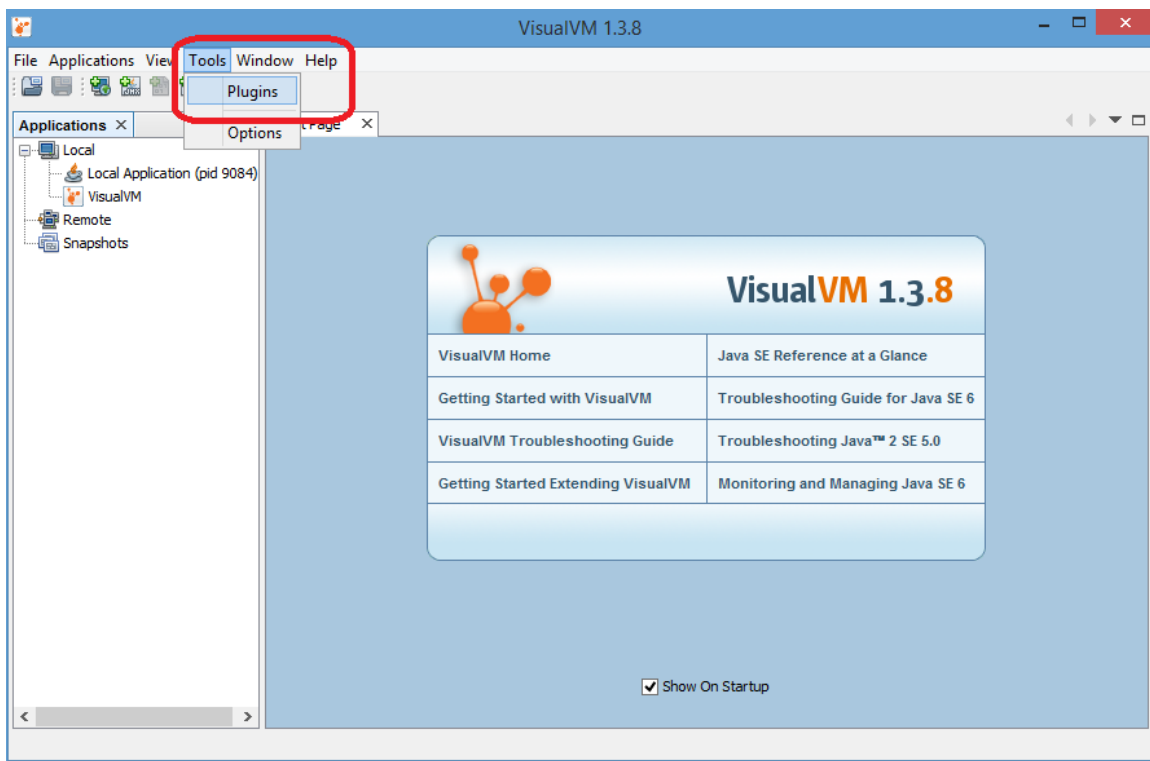
```
192.168.1.139 - PuTTY
login as: bagri
Using keyboard-interactive authentication.
Password:
Last login: Fri Mar 25 09:00:37 2016 from 192.168.1.87
bagri@linbox:~>
bagri@linbox:~> cd apps/bagri/bagri-0.6.1-SNAPSHOT/bin/
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> source $HOME/.bashrc
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> ./bgadmin.sh stop
stopping all nodes
stopping process (pid 20596)...
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
```


6. Configure Graphical User Interface (optional)

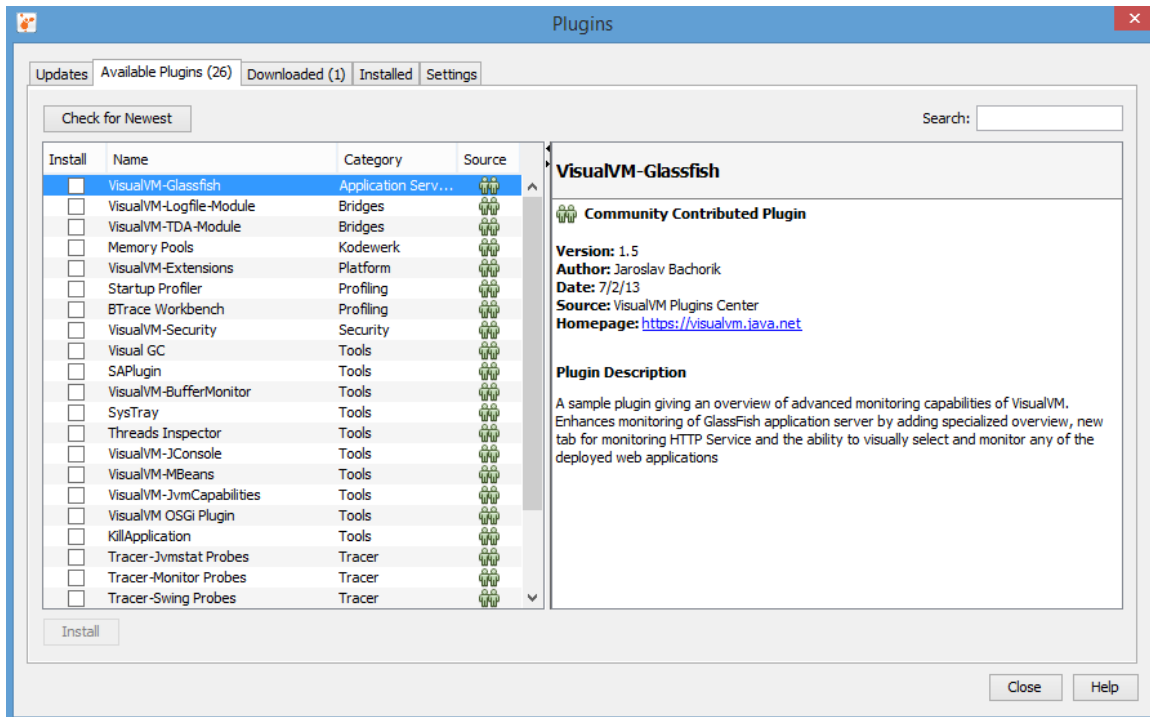
In order to see information about running Bagri servers you can use a graphical user interface applications provided by the 3rd party software such as VisualVM or any other JMX console. This section present steps to perform in order to manage and monitor system's behavior in the VisualVM app.

To Configure GUI on Windows platform

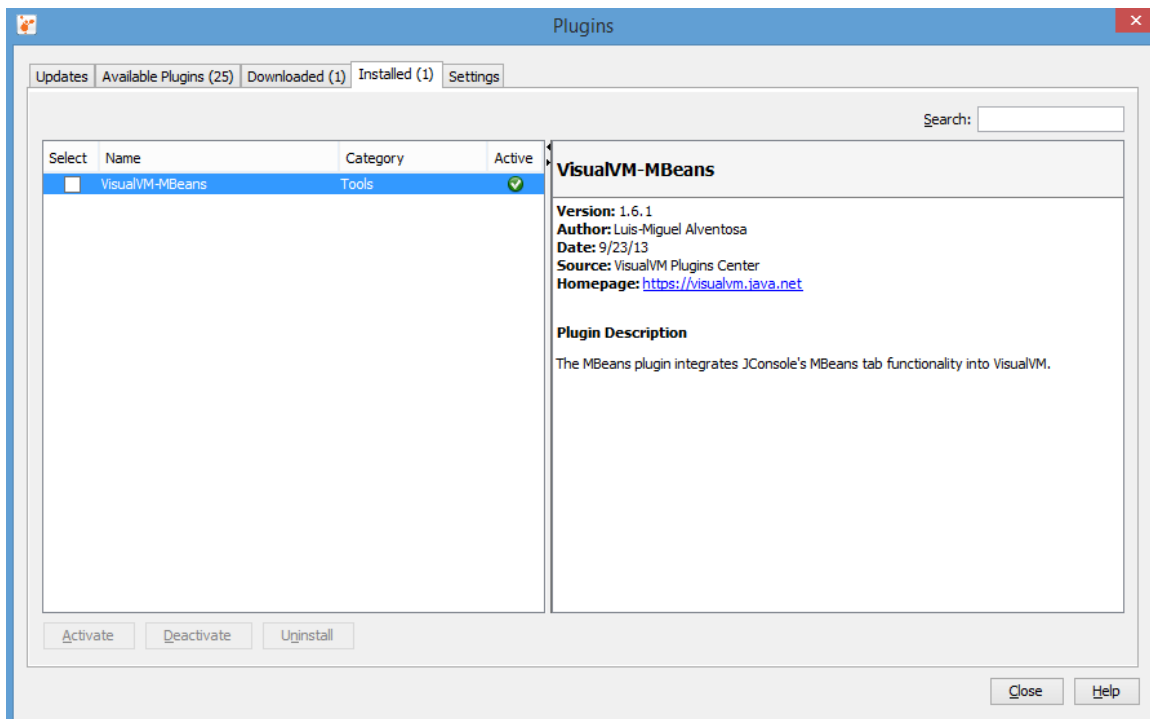
1. Download [VisualVM](#) and install it on your computer.
2. Navigate to VisualVM installation directory → visualvm_138 → bin and run visualvm.exe
3. Install VisualVM-MBeans plugin:
 - In the main VisualVM menu navigate to **Tools→Plugins**.



- In the **Plugins** dialog that opens, select **Available Plugins** tab.



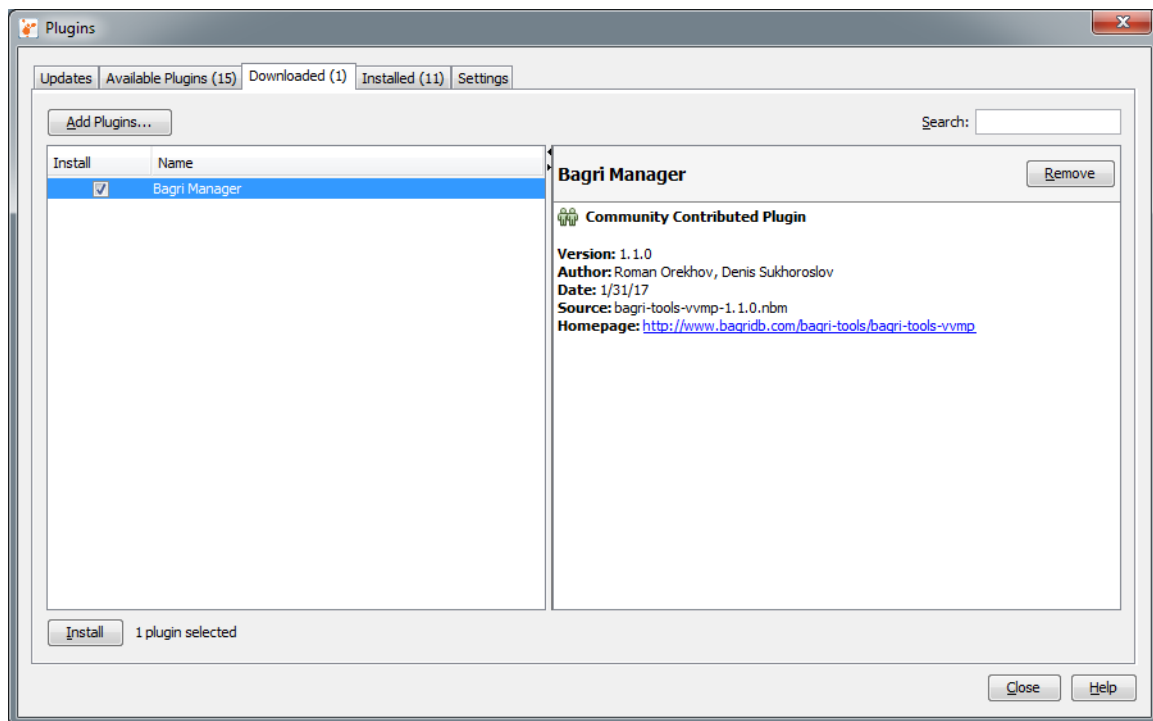
- Locate **VisualVM-MBeans** plugin and click **Install** to start the installation process.
- Follow the instructions of the installation wizard. Once it is finished, select **Installed** tab and check whether **VisualVM-MBeans** plugin occurred in the list of the installed plugins.



4. Install **Bagri Manager** plugin:

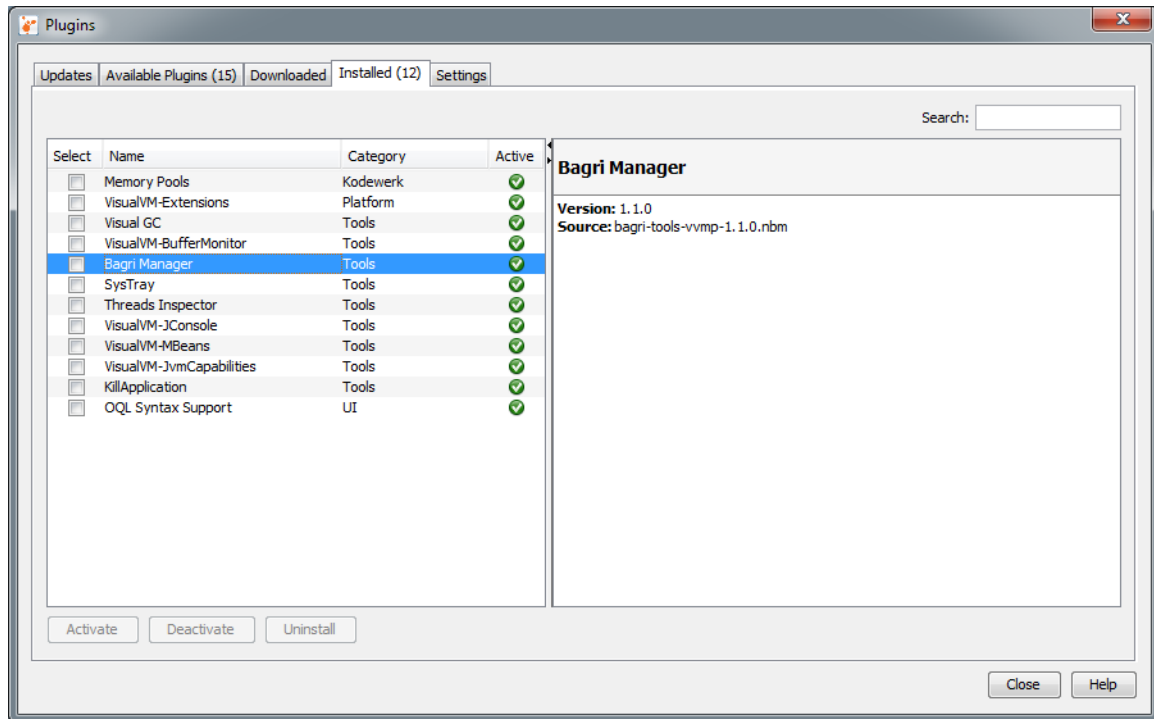
- In the main VisualVM menu navigate to **Tools→Plugins**.
- In the **Plugins** dialog that opens, select **Downloaded** tab and then click the **Add Plugins...** button.
- Navigate to Bagri installation folder → distr, select bagri-tools-vvmp-1.1.0.nbm and click **Open**.

Bagri Manager plugin will be added to the list of the plugins available for installation.



- Select **Bagri Manager** plugin and click **Install**.
- Follow the instructions of the installation wizard. On the last step of the installation, you will be asked to restart the system. Choose 'Restart Now' and click **Finish**.

Once **VisualVM** is restarted, check whether **Bagri Manager** plugin occurred in the list of the installed plugins:



7. Appendix A: Configuration Files

7.1. Configuration file access.xml

The `access.xml` file contains two sections specifying roles and users who suppose to work with the system.

The table below list all roles required to manage the product. They are divided into two types: functional roles which are required to manage various parts of the system functionality and composite roles which are constructed as a combination of some functional roles.

Role Name	Description
Functional Roles	
DataFormatManagement	Manages plugins to work with external document formats.
DataStoreManagement	Manages plugins connecting to external document stores.
LibraryManagement	Allows to manage external Java libraries which contain functions and triggers
ModuleManagement	Allows to manage external XQuery modules that contains supporting functions
NodeManagement	Allows to define configuration templatesfor nodes
SchemaManagement	Allows to manage schemas and their components such as collections, models, indices, triggers, etc.
RolesManagement	Allows tomanage roles that are used to work with the system
UserManagement	Allows to manage users who will access the system
Composite Roles	
AdminRole	This role combines all functional roles that are listed above
GuestRole	This role allows user to read from any schema of the system

By default, the product is delivered with two composite roles: AdminRole and GuestRole. However, additional composite roles can be created by using available functional roles, in

case it is required for some reason. The new roles should have the same format and properties as it is shown in the table below:

Property	Description
name	Role name
version	Role version
createdAt	The date and time when role was created.
createdBy	The user who has created a role.
permissions	The set of permissions assigned the role. Possible permission values are: read/modify/execute. Permission values are separated by space character.
resource	Permissions are granted on specific system resources. The resource names are specified in JMX Query syntax.
includedRoles	Other roles those are included in this composite role.
description	Textual definition of a role.

The example below shows the format for the role:

```
<role name="GuestRole">
  <version>3</version>
  <createdAt>2016-03-27T00:36:16.320+04:00</createdAt>
  <createdBy>admin</createdBy>
  <permissions>
    <permission resource="com.bagri.db:name=*,type=Schema">read</permission>
  </permissions>
  <includedRoles></includedRoles>
  <description>Description of the role</description>
</role>
```

The second section of the file contains a list of user accounts who will have an access to the system. Every user has the following parameters:

Property	Description
login	A name for an account.
active	Status of an account.

Property	Description
version	Account version
createdAt	Date and time when an account was created.
createdBy	The user who has created an account.
permissions	The set of permissions assigned the account. Possible permission values are: read/modify/execute. Permission values are separated with space character.
resource	Permissions are granted on specific system resources. The resource names are specified in JMX Query syntax.
includedRoles	Roles that are assigned to the account.
password	Encrypted password that is used by account to work with the system.

The example below shows the format for the user account:

```
<user login="admin" active="true">
  <version>2</version>
  <createdAt>2016-03-29T01:12:30.559+04:00</createdAt>
  <createdBy>admin</createdBy>
  <permissions>
    <permission resource="com.bagri.db:name=*,type=Schema">read
modify</permission>
  </permissions>
  <includedRoles>AdminRole</includedRoles>
  <password>5f4dcc3b5aa765d61d8327deb882cf99</password>
</user>
```

By default, the `access.xml` configuration file contains two users: **admin** and **guest**. If additional users are required, they can be created and assigned with the necessary roles following the format described above.

7.2. Configuration file config.xml

The file `config.xml` consists of several parameter sections which specify various aspects of the system behavior.

Nodes: Node template is a set options that can be used by cache servers. Each node has the following paramters:

Property	Description
----------	-------------

Property	Description
name	The node template name
version	Node version
createdAt	Date and time when node was created
createdBy	The user who has created a node
options	The list of node options in key/value format

The example below shows the structure and properties of a node template:

```
<node name="cache">
  <version>1</version>
  <createdAt>2016-03-24T17:54:42.221+04:00</createdAt>
  <createdBy>admin</createdBy>
  <options>
    <entry name="bdb.cluster.node.schemas">default</entry>
    <entry name="bdb.cluster.node.role">server</entry>
  </options>
</node>
```

Modules: Module is a set of XQuery functions. Each module has the following parameters:

Property	Description
name	The module name
version	Module version
createdAt	Date and time when module was created
createdBy	The user who has created a module
fileName	An absolute or relative path to the module file
description	Textual definition of the module
prefix	Prefix associated with the module namespace
namespace	Module namespace
enabled	The module can be enabled or not

The example below shows the structure and properties of a module:

```
<module name="trigger_module">
  <version>1</version>
  <createdAt>2016-03-10T13:36:26.965+03:00</createdAt>
```



```
<createdBy>admin</createdBy>
<fileName>../data/tpox/sample_triggers.xq</fileName>
<description>The Trigger Sample Module</description>
<prefix>bgdb</prefix>
<namespace>http://bagridb.com/bdb</namespace>
<enabled>true</enabled>
</module>
```

Libraries: Library is a Java jar file containing supporting functions and/or trigger implementations. Each library has the following parameters:

Property	Description
name	The library name
version	Library version
createdAt	Date and time when library was created
createdBy	The user who has created a library
description	Textual definition of the library
enabled	The library can be enabled or not
functions	A list of function declarations contained in the library. Every function has the following list of parameters:
className	Java class name
method	Method name
result.type	Java type of returning value
result.cardinality	The number of returning elements. Possible values are: one, one_or_more, zero_or_one, zero_or_more;
prefix	Function namespace prefix
parameters	A list of parameters declared in the function. Every parameter has the following list of attributes:
name	Parameter name
type	Parameter type
cardinality	The number of parameter elements. Possible values are the same as for result

The example below shows the structure and the properties of a library:

```
<library name="java_library">
  <version>5</version>
  <createdAt>2016-03-21T16:17:20.542+03:00</createdAt>
  <createdBy>admin</createdBy>
  <description>The Standard Java Extension Library</description>
  <enabled>true</enabled>
  <functions>
    <function>
      <className>java.lang.Math</className>
      <method>max</method>
      <result type="long" cardinality="one"/>
      <prefix>math</prefix>
      <parameters>
        <parameter name="arg0" type="long" cardinality="one"/>
        <parameter name="arg1" type="long" cardinality="one"/>
      </parameters>
    </function>
  </functions>
</library>
```

DataFormats: DataFormat is a plugin implemented in Java. It contains classes for parsing and producing text documents in some external format, not handled by the system yet. Each data format has the following parameters:

Property	Description
name	The data format name
version	Data format plugin version
createdAt	Date and time when data format was created
createdBy	The user who has created a data format
description	Textual definition of the data format
enabled	The data format can be enabled or disabled
parserClass	The name of Java class implementing ContentParser interface
builderClass	The name of Java class implementing ContentBuilder interface
extensions	File extensions which are registered with this data format
properties	An optional list of properties which will be passed to the parser/builder classes at initialization phase

The example below shows the structure and properties of a data format:

```
<dataFormat name="JSON">
```

```
<version>1</version>
<createdAt>2016-05-31T16:17:20.542+03:00</createdAt>
<createdBy>admin</createdBy>
<description>JSON Parser/Builder</description>
<parserClass>com.bagri.core.server.api.df.json.JsonApiParser</parserClass>
<builderClass>com.bagri.core.server.api.df.json.JsonBuilder</builderClass>
<enabled>true</enabled>
<extensions>json</extensions>
<properties/>
</dataFormat>
```

DataStores: DataStore is another plugin implementing connectivity with external document store. In this way developers can connect the system with external document storages like Hadoop, Mongo, etc. The data store plugin defined with the following parameters:

Property	Description
name	The data store name
version	Data store plugin version
createdAt	Date and time when data store was created
createdBy	The user who has created a data store
description	Textual definition of external data store
enabled	The data format can be enabled or disabled
storeClass	The name of Java class implementing DocumentStore interface
properties	The list of connectivity properties which will be passed to the data store at initialization phase

The example below shows the structure and properties of a data format:

```
<dataStore name="File">
  <version>1</version>
  <createdAt>2015-05-31T16:17:20.542+03:00</createdAt>
  <createdBy>admin</createdBy>
  <description>Standard File store</description>
  <enabled>true</enabled>
  <storeClass>com.bagri.server.hazelcast.store.FileDocumentCacheStore</storeClass>
  <properties>
    <entry name="bdb.schema.store.data.path">../data</entry>
    <entry name="bdb.schema.store.read-only">false</entry>
  </properties>
</dataStore>
```

Schemas: the section lists the schemas that are registered in the system. By default, the product is delivered with four pre-defined schemas:

- default
- TPoX
- XMark
- TPoX-j

Each schema has the following parameters:

Property	Description
name	Schema name
active	Schema status. The value can be true or false.
version	Schema version
createdAt	Date and time when schema was created
createdBy	The user who has created a schema
description	Textual definition of the schema
properties	A list of schema properties specified as key/value pairs. For the full list of schema properties, please refer to the tables below

The following example shows the structure and properties of a schema:

```
<schema name="TPoX" active="true">
  <version>1</version>
  <createdAt>20146-03-21T14:40:58.096+04:00</createdAt>
  <createdBy>admin</createdBy>
  <description>TPoX: schema for TPoX-related tests</description>
  <properties>
    <entry name="bdb.schema.ports.first">10000</entry>
    <entry name="bdb.schema.ports.last">10100</entry>
    <entry name="bdb.schema.members">localhost</entry>
    <entry name="bdb.schema.store.data.path">../data/tpox</entry>
    <entry name="bdb.schema.store.type">File</entry>
    <entry name="bdb.schema.format.default">XML</entry>
    .....
    <entry name="xqj.schema.baseUri">file:///../data/tpox/</entry>
    <entry name="xqj.schema.orderingMode">2</entry>
    <entry name="xqj.schema.queryLanguageTypeAndVersion">1</entry>
    <entry name="xqj.schema.defaultCollationUri"></entry>
    .....
  </properties>
</collections/>
```

```
<fragments/>
<indexes/>
<resources/>
<triggers/>
</schema>
```

After the properties subsection every schema contains another five parameter blocks:

Collections: this subsection describes collections registered in the schema. Collections are used to group some schema documents together and run XQueries against the group of documents. Each collection has the following parameters:

Property	Description
name	The collection name
version	Collection version
createdAt	Date and time when collection was created
createdBy	The user who has created a collection
docType	XPath for default collection types, optional
description	Textual definition of the collection
enabled	The collection can be enabled or not

The example below shows the structure and the properties of a collection:

```
<collection id="3" name="CLN_Order">
  <version>1</version>
  <createdAt>2016-03-20T01:01:26.965+03:00</createdAt>
  <createdBy>admin</createdBy>
  <docType>/({http://www.fixprotocol.org/FIXML-4-4}FIXML</docType>
  <description>All order documents</description>
  <enabled>true</enabled>
</collection>
```

Fragments: this subsection lists fragments that specify repeating data structures in schema documents. Each fragment has the following parameters:

Property	Description
name	The fragment name
version	Fragment version
createdAt	Date and time when a fragment was created

Property	Description
createdBy	The user who has created a fragment
docType	XPath to the corresponding document's root
path	XPath to the repetitive fragment root
description	Textual definition of the fragment
enabled	The fragment can be enabled or not

The example below shows the structure and the properties of a fragment:

```
<fragment name="FRA_Categories_Category">
  <version>1</version>
  <createdAt>2016-03-02T19:05:26.965+03:00</createdAt>
  <createdBy>admin</createdBy>
  <docType>/site</docType>
  <path>/site/categories/category</path>
  <description>Auction Categories</description>
  <enabled>true</enabled>
</fragment>
```

Indexes: the section lists indices registered in schema to speed up queries against schema documents. Each index has the following parameters:

Property	Description
name	The index name
version	Index version
createdAt	Date and time when an index was created
createdBy	The user who has created an index
docType	XPath to the corresponding document's root
path	XPath to the indexed value
dataType	Indexed value data type
caseSensitive	Is index case sensitive or not. Possible values are true/false
range	Is index supports ordering. Possible values are true/false
unique	Is index unique. Possible values are true/false

Property	Description
description	Textual definition of the index
enabled	The index can be enabled or not

The example below shows the structure and the properties of an index:

```
<index name="IDX_Customer_id">
  <version>1</version>
  <createdAt>2016-03-25T13:36:26.965+03:00</createdAt>
  <createdBy>admin</createdBy>
  <docType>/{http://tpox-benchmark.com/custacc}Customer</docType>
  <path>/{http://tpox-benchmark.com/custacc}Customer/@id</path>
  <dataType xmlns:xs="http://www.w3.org/2001/XMLSchema">xs:string</dataType>
  <caseSensitive>true</caseSensitive>
  <range>false</range>
  <unique>true</unique>
  <description>Customer id</description>
  <enabled>true</enabled>
</index>
```

Resources: the section lists REST resources registered in schema to allow access to Schema functionality via HTTP. Each resource has the following parameters:

Property	Description
name	The resource name
version	Resource version
createdAt	Date and time when resource was created
createdBy	The user who has created the resource
path	relative path used in HTTP requests to access resource
module	Name of the XQuery module implementing resource functionality
description	Textual definition of the resource
enabled	The resource can be enabled or not

The example below shows the structure and the properties of a resource:

```
<resource name="tpox">
  <version>1</version>
  <createdAt>2016-10-04T13:36:26.965+03:00</createdAt>
  <createdBy>admin</createdBy>
  <path>/tpox</path>
```

```
<module>rest_module</module>
<description>TPoX resource exposed via REST</description>
<enabled>true</enabled>
</resource>
```

Triggers: the section lists triggers that are defined in the schema. The triggers can be applied at the moment of creation, amendment or deletion of schema document. Triggers can be implemented as Java classes and referenced from extension library, or as XQuery function and referenced from XQuery server module. Each trigger has the following parameters:

Property	Description
name	The trigger name
version	Trigger version
createdAt	Date and time when a trigger was created
createdBy	The user who has created a trigger
docType	XPath to the corresponding document's root
synchronous	Trigger can be fired synchronously (<code>true</code>) or asynchronously (<code>false</code>)
enabled	Trigger can be enabled or not
index	An order at which trigger will be fired
actions	The list of action when trigger will be fired. Every action has the following attributes:
order	When the trigger will be fired, right before or after the scope
scope	A moment in document lifecycle when an event takes place. Possible values are: <code>insert</code> , <code>update</code> or <code>delete</code>
library	For Java triggers: a reference to registered library containing Java trigger implementation
className	For Java triggers: Full Java class name of the class implementing trigger
module	For XQuery triggers: a reference to registered module containing XQuery trigger implementation
function	For XQuery triggers: The name of XQuery function implementing trigger

The example below shows the structure and properties of a Java trigger:


```
<trigger xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:javatrigger">
  <version>1</version>
  <createdAt>2016-03-27T02:25:21.887+03:00</createdAt>
  <createdBy>admin</createdBy>
  <docType>/{http://tpox-benchmark.com/security}Security</docType>
  <synchronous>true</synchronous>
  <enabled>true</enabled>
  <index>1</index>
  <actions>
    <action order="after" scope="insert"/>
  </actions>
  <library>trigger_library</library>
  <className>com.bagri.samples.ext.SecurityTrigger</className>
</trigger>
```

To adjust the schema behavior user can specify elements explained above according to his/her requirements. Additionally, the user can setup properties which are used in schemas. The tables below explain all the properties with their default or sample values.

Schema configuration (BDB) properties:

Property name	Mandatory (Y/N)	Default or Sample Value	Description
bdb.schema.password	Y		encrypted (?) schema password
bdb.schema.store.enabled	N	false	enable/disable schema persistence
bdb.schema.store.tx.buffer.size		2048	schema transaction journal buffer size
bdb.schema.store.type		File	The name of the plugin implementing data store for the schema.
bdb.schema.store.data.path	Y	../data	
bdb.schema.format.default		XML	The name of the data format plugin which will perform document parsing and producing
bdb.schema.population.size		1	schema population size

Property name	Mandatory (Y/N)	Default or Sample Value	Description
			(node count)
bdb.schema.population.buffer.size		1000000	schema catalog buffer size
bdb.schema.buffer.size		64	send/receive packet size between client and server, in Kb
bdb.schema.members		localhost	Addresses of schema nodes
bdb.schema.ports.first		20000	schema cluster port range start
bdb.schema.ports.last		20010	schema cluster port range end
bdb.schema.thread.pool		32	schema cluster operation thread pool size
bdb.schema.partition.pool		32	schema cluster partition thread pool size
bdb.schema.partition.count		271	schema cluster partition count
bdb.schema.transaction.timeout		60000	schema transaction timeout (ms)
bdb.schema.health.threshold.low		25	lower HealthMonitor bound
bdb.schema.health.threshold.high		0	upper HealthMonitor bound
bdb.schema.query.parallel		true	to perform queries on partitions in parallel or not
bdb.schema.data.backup.async		1	number of schema documents asynchronous backups
bdb.schema.data.backup.sync		0	number of schema documents synchronous

Property name	Mandatory (Y/N)	Default or Sample Value	Description
			backups
bdb.schema.data.backup.read		false	allow to read schema documents from backup replica
bdb.schema.dict.backup.async		1	number of schema dictionaries asynchronous backups
bdb.schema.dict.backup.sync		0	number of schema dictionaries synchronous backups
bdb.schema.dict.backup.read		true	allow to read schema dictionaries from backup replica
bdb.schema.query.backup.async		0	number of schema cached query asynchronous backups
bdb.schema.query.backup.sync		0	number of schema cached query synchronous backups
bdb.schema.query.backup.read		true	allow to read schema cached queries from backup replica
bdb.schema.trans.backup.async		0	number of schema transactions asynchronous backups
bdb.schema.trans.backup.sync		1	number of schema transactions synchronous backups
bdb.schema.trans.backup.read		false	allow to read schema transactions from backup replica

A schema can also specify properties for the DataFormat and DataStore plugins referenced by the schema. In this way any schema can specify its own property values for one common plugin.

XQuery processing (XQJ) properties:

Property name	Mandatory (Y/N)	Default or Sample Value	Description
xqj.schema.baseUri	Y	/opt/data	Schema base URI
xqj.schema.orderingMode	N	2	Ordering mode
xqj.schema.queryLanguageTypeAndVersion	N	1	XQuery language version
xqj.schema.bindingMode	N	0	Binding mode
xqj.schema.boundarySpacePolicy	N	1	Boundary-space policy
xqj.schema.scrollability	N	1	Scrollability of the result sequences
xqj.schema.holdability	N	2	Holdability of the result sequences
xqj.schema.copyNamespacesModePreserve	N	1	Copy-namespaces preserve mode
xqj.schema.queryTimeout	N	0	XQuery timeout
xqj.schema.defaultFunctionNamespace	N	http://www.w3.org/2005/xpath-functions	Default function namespace
xqj.schema.defaultElementTypeNamespace	N	http://www.w3.org/2001/XMLSchema	Default element/type namespace
xqj.schema.copyNamespacesModelInherit	N	1	Copy-namespaces inherit mode
xqj.schema.defaultOrderForEmptySequences	N	2	Default order for empty sequences
xqj.schema.defaultCollationUri	N	http://www.	Default collation

Property name	Mandatory (Y/N)	Default or Sample Value	Description
		w3.org/2005 /xpath-functions/collation/code point	
xqj.schema.constructionMode	N	1	Construction mode

7.3. Configuration file admin.properties

The `admin.properties` file consists of the parameters that are used when Administrative server is launched.

Property name	Mandatory (Y/N)	Default or Sample Value	Description
bdb.cluster.admin.port	N	3330	administrative server JMX port
bdb.cluster.port	N	3331	administrative server access port
bdb.cluster.thread.pool	N	8	administrative server thread pool size
bdb.cluster.members	N	localhost	administrative cluster known nodes
bdb.cluster.login	N	admin	login to connect to Data servers
bdb.client.fetchSize	N	1000	query page size
bdb.rest.jmx	N	true	to start embedded REST server (jetty) MBeans or not
bdb.rest.port	N	3030	REST server HTTP port
bdb.rest.auth.port	N	3443	REST server HTTPS port

7.4. Configuration files <profile_name>.properties

The profile configuration file `<profile_name>.properties` consists of the properties that are used when Data server is launched.

Property name	Mandatory (Y/N)	Default or Sample Value	Description
bdb.cluster.port	N	3331	administrative cluster access port
bdb.cluster.thread.pool	N	8	administrative cluster client thread pool size
bdb.cluster.members	N	localhost	administrative cluster known nodes
bdb.cluster.node.schemas	Y	default	schemas deployed on Data servers

The system delivered with four demo profiles: first.properties, second.properties, third.properties and fourth.properties. All of them have the same default property set except the schema names deployed on the profile.

8. Related Documentation

The table below lists all other documents available to support Bagri:

Document	Description

Also, in case of any questions or to get more info please visit [Bagri official web site](#) and/or [Bagri mail group](#).