Installation and Configuration Guide

Version: 1.2.1
3/10/2018

Table of Contents

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, Ignite 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 requirements for computer where Bagri DB is going to beinstalled. Refer to the following sections for detailed information:

- Hardware Requirements
- Software Requirements
- Deployment Options

2.1. Hardware Requirements

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

Hardware Component	Minimum	Recommended
Processor	No special requirements	Intel Core i3..i7, 2..4GHz
RAM	2 GB	4 – 8 GB per node, depending on the database size **NOTE**: Before running the Product, please make sure that some rough calculation of the required memory has been performed. The default value for one node set to 4 GB, if you need to change it, you can manually specify required value in the configuration file. Please refer to the Initial Configuration to run the Product section for more details.
Disk Space	100 MB for initial installation	Disk space required for Bagri to function properly depends on the database size; an average size for each node can be from 10 to 100 GB or more.

2.2. Software Requirements

The table below specifies minimum software requirements for Bagri:

Component	Requirements
Operating system	• Desktop (Client) OS: Microsoft Windows 7 and above SUSE Linux v.11 SP4 openSUSE Tumbleweed OS X Maverick Any other *nix distribution with Java installed

Component	Requirements
	• 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	• Java 1.7 or 1.8 **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 platform. • VisualVM 1.3.6 or higher **NOTE:** VisulaVMor other JMX console (like JConsole, which is a part of JDK installation) is an optional tool which can be used to monitor and manage system behavior.

2.3. Deployment Options

This section provides recommendations on how BagriDB can be deployed. 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 or database 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	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. Such deployment allows to achieve maximum performance.
client-server application with distributed document database	This option can be used for processing greater amounts of documents with several schemas applied. Clients can access schemas deployed in dedicated distributed cache clusters via provided XQJ driver or via Bagri Java/.NET client API or via Bagri REST interface. System scaling can be performed dynamically by adding new nodes to existing schema clusters which does not require system downtime.

Option	Description
	Such deployment is best suited for on-line processing of terabytes of data with memory-access speed.

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-1.2.1.zip** for Windows based machine;
- **bagri-1.2.1.tar.gz** for Linux based machine;

where digits represent release version of the product.

The archive consists of the following folders:

Folder name	Description
`bin`	This folder contains shell scripts to start Bagri Administrative and/or Data servers: • `bgadmin.cmd` – launches Administrative server on Windows platform • `bgadmin.conf` – supporting configuration file to run Administrative server on Linux platform • `bgadmin.sh` – launches Administrative server on Linux platform • `bgcache.cmd` – launches Data servers on Windows platform • `bgcache.conf` – supporting configuration file to run Data server on Linux platform • `bgcashe.sh` – launches Data server on Linux platform • `bgstop.cmd` – stops Data server(s) on Windows platform
`config`	This folder contains sample Bagri profiles and their configuration settings. The folder consists of the following files: • `access.xml` – contains basic access settings. • `config.xml` – contains configuration settings for the profiles. • `first.properties`, `second.properties`, etc. – initial profile parameters. • `logging.xml`, `hz-logging.xml` – logging configurations. Please refer to the Appendix A: Configuration Files section below for the detailed overview of all the configuration parameters.
`data`	This folder contains subfolders with sample documents that can be used for the systeme valuation: • `json` – contains sample JSON documents • `tpox` – contains sample TPoX documents and queries • `xmark` – contains sample XMark document and queries
`distr`	This folder contains redistributable client artifacts i.e. JSR-225 XQJ driver, Bagri plugin for VisualVM console and Bagri assembly for .NET framework
`docs`	This folder contains Bagri documentation: • 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

Folder name	Description
`samples`	This folder contains sample programs which demoes 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 set up Bagri environment and start using the system.

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 Administartive server host, for instance:

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

   - Save the file

6. Specify network settings to be used by the choosen 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>
        .........
```

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`". Type `change directory` command and path to the folder where `bgcache.cmd` is located and press **Enter**.

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

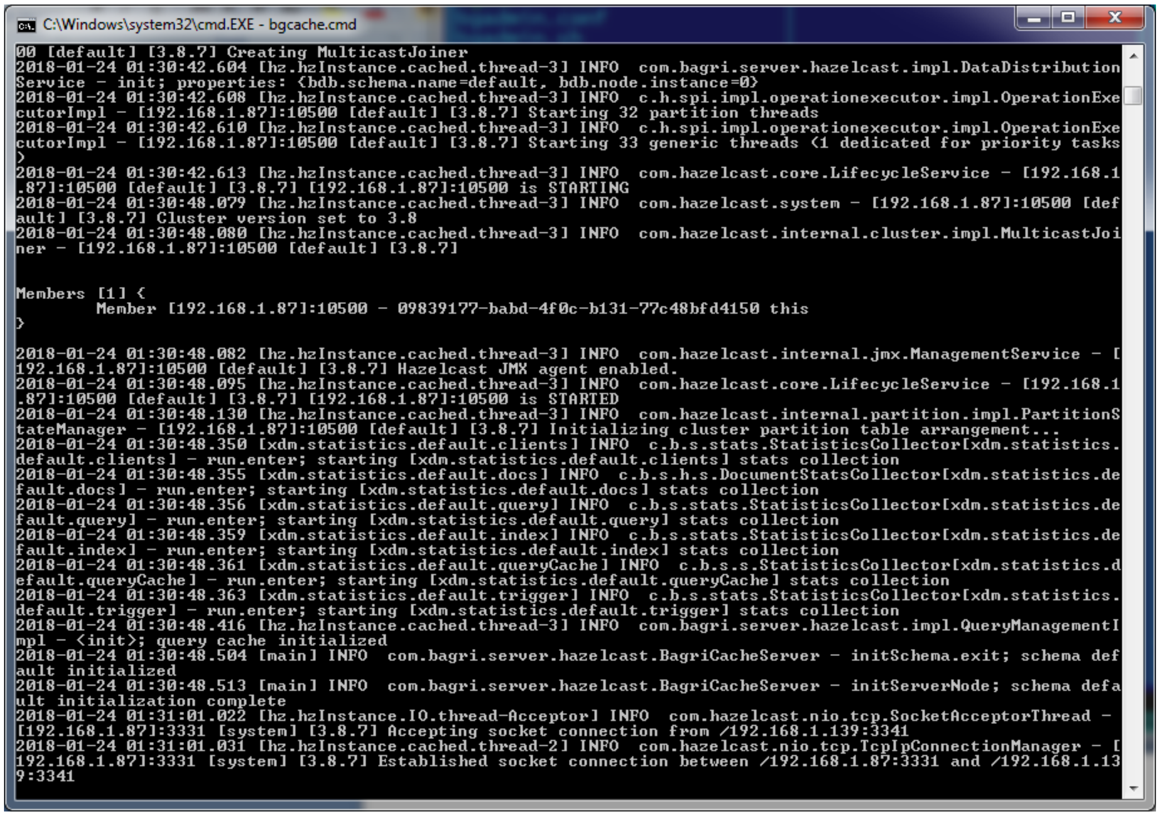
2. Type the following command to start Bagri node:

   ```
   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 schema cluster. This file must be 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 then application use 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 successful launch of the application, the system will log output to console:



```
00 [default] [3.8.7] Creating MulticastJoiner
2018-01-24 01:30:42.604 [hz.hzInstance.cached.thread-3] INFO  com.bagri.server.hazelcast.impl.DataDistribution
Service - init; properties: {bdb.schema.name=default, bdb.node.instance=0}
2018-01-24 01:30:42.608 [hz.hzInstance.cached.thread-3] INFO  c.h.spi.impl.operationexecutor.impl.OperationExe
cutorImpl - [192.168.1.87]:10500 [default] [3.8.7] Starting 32 partition threads
2018-01-24 01:30:42.610 [hz.hzInstance.cached.thread-3] INFO  c.h.spi.impl.operationexecutor.impl.OperationExe
cutorImpl - [192.168.1.87]:10500 [default] [3.8.7] Starting 33 generic threads (1 dedicated for priority tasks
)
2018-01-24 01:30:42.613 [hz.hzInstance.cached.thread-3] INFO  com.hazelcast.core.LifecycleService - [192.168.1
.87]:10500 [default] [3.8.7] [192.168.1.87]:10500 is STARTING
2018-01-24 01:30:48.079 [hz.hzInstance.cached.thread-3] INFO  com.hazelcast.system - [192.168.1.87]:10500 [def
ault] [3.8.7] Cluster version set to 3.8
2018-01-24 01:30:48.080 [hz.hzInstance.cached.thread-3] INFO  com.hazelcast.internal.cluster.impl.MulticastJoi
ner - [192.168.1.87]:10500 [default] [3.8.7]


Members [1] {
        Member [192.168.1.87]:10500 - 09839177-babd-4f0c-b131-77c48bfd4150 this
}

2018-01-24 01:30:48.082 [hz.hzInstance.cached.thread-3] INFO  com.hazelcast.internal.jmx.ManagementService - [
192.168.1.87]:10500 [default] [3.8.7] Hazelcast JMX agent enabled.
2018-01-24 01:30:48.095 [hz.hzInstance.cached.thread-3] INFO  com.hazelcast.core.LifecycleService - [192.168.1
.87]:10500 [default] [3.8.7] [192.168.1.87]:10500 is STARTED
2018-01-24 01:30:48.130 [hz.hzInstance.cached.thread-3] INFO  com.hazelcast.internal.partition.impl.PartitionS
tateManager - [192.168.1.87]:10500 [default] [3.8.7] Initializing cluster partition table arrangement...
2018-01-24 01:30:48.350 [xdm.statistics.default.clients] INFO  c.b.s.stats.StatisticsCollector[xdm.statistics.
default.clients] - run.enter; starting [xdm.statistics.default.clients] stats collection
2018-01-24 01:30:48.355 [xdm.statistics.default.docs] INFO  c.b.s.h.s.DocumentStatsCollector[xdm.statistics.de
fault.docs] - run.enter; starting [xdm.statistics.default.docs] stats collection
2018-01-24 01:30:48.356 [xdm.statistics.default.query] INFO  c.b.s.stats.StatisticsCollector[xdm.statistics.de
fault.query] - run.enter; starting [xdm.statistics.default.query] stats collection
2018-01-24 01:30:48.359 [xdm.statistics.default.index] INFO  c.b.s.stats.StatisticsCollector[xdm.statistics.de
fault.index] - run.enter; starting [xdm.statistics.default.index] stats collection
2018-01-24 01:30:48.361 [xdm.statistics.default.queryCache] INFO  c.b.s.s.StatisticsCollector[xdm.statistics.d
efault.queryCache] - run.enter; starting [xdm.statistics.default.queryCache] stats collection
2018-01-24 01:30:48.363 [xdm.statistics.default.trigger] INFO  c.b.s.stats.StatisticsCollector[xdm.statistics.
default.trigger] - run.enter; starting [xdm.statistics.default.trigger] stats collection
2018-01-24 01:30:48.416 [hz.hzInstance.cached.thread-3] INFO  com.bagri.server.hazelcast.impl.QueryManagementI
mpl - <init>; query cache initialized
2018-01-24 01:30:48.504 [main] INFO  com.bagri.server.hazelcast.BagriCacheServer - initSchema.exit; schema def
ault initialized
2018-01-24 01:30:48.513 [main] INFO  com.bagri.server.hazelcast.BagriCacheServer - initServerNode; schema defa
ult initialization complete
2018-01-24 01:31:01.022 [hz.hzInstance.IO.thread-Acceptor] INFO  com.hazelcast.nio.tcp.SocketAcceptorThread -
[192.168.1.87]:3331 [system] [3.8.7] Accepting socket connection from /192.168.1.139:3341
2018-01-24 01:31:01.031 [hz.hzInstance.cached.thread-2] INFO  com.hazelcast.nio.tcp.TcpIpConnectionManager - [
192.168.1.87]:3331 [system] [3.8.7] Established socket connection between /192.168.1.87:3331 and /192.168.1.13
9:3341
```

Once the system is started, it writes all important events to corresponding log file 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`.

4. To stop Bagri node just hit `Ctrl + C` in the console window.

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 `bgcache.sh` is located and press **Enter**.

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

2. Type the following command:

   ```
   >./bgcache.sh start <profile-name> <node-number>
   ```

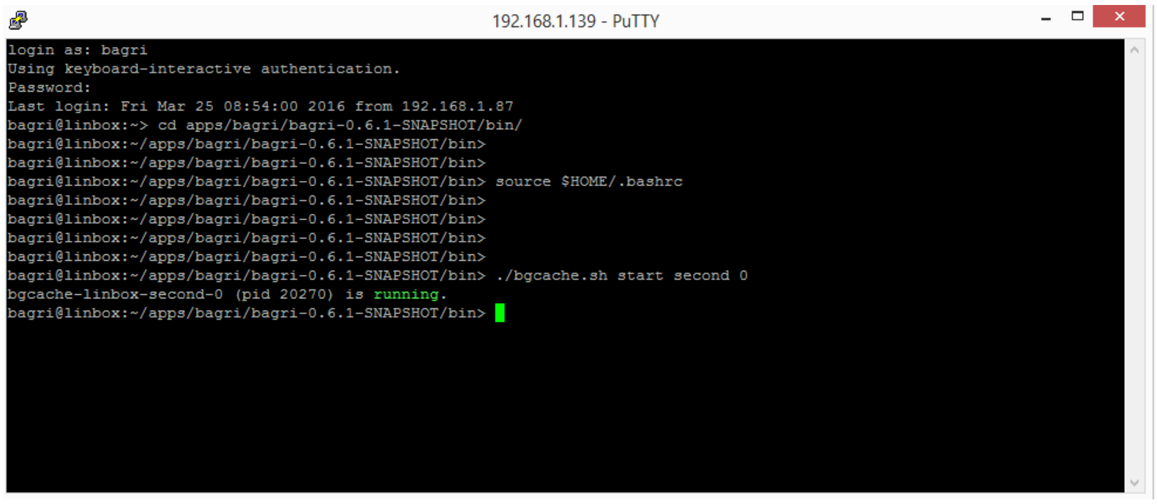
   The command parameters are:

   - `bgcache.sh`: name of the shell script to launch Bagri Data server.
   - `start`: the command to start new node instance
   - `profile-name`: name of the 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 will use 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 will use default value equal to `0`.

3. On successful launch of the system, the script will display message that the node has been started properly.

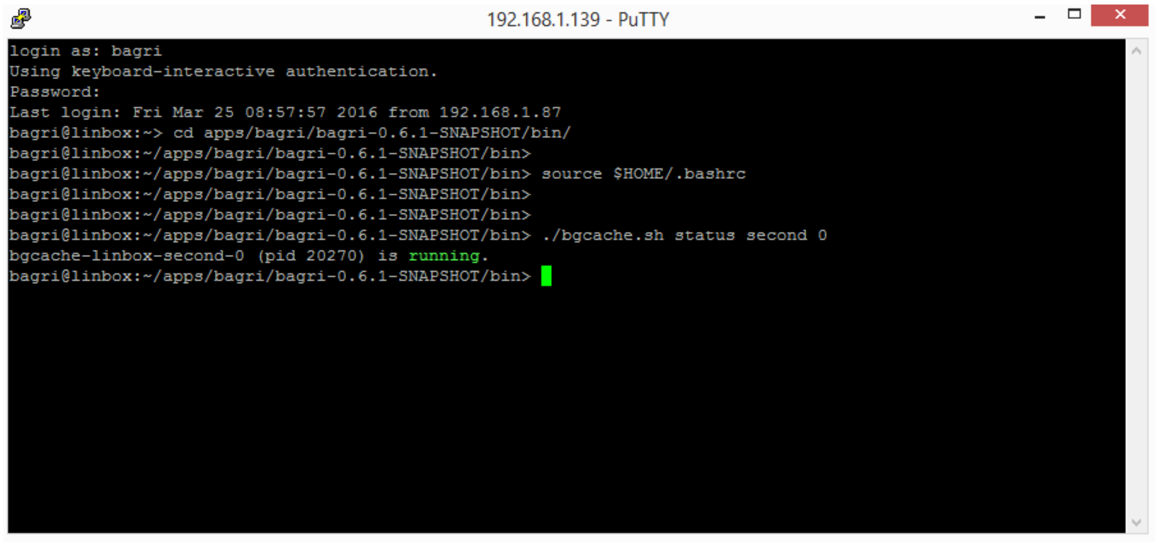


```
login as: bagri
Using keyboard-interactive authentication.
Password:
Last login: Fri Mar 25 08:54:00 2016 from 192.168.1.87
bagri@linbox:~> cd apps/bagri/bagri-0.6.1-SNAPSHOT/bin/
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin>
bagri@linbox:~/apps/bagri/bagri-0.6.1-SNAPSHOT/bin> 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>
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 to corresponding log file 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:

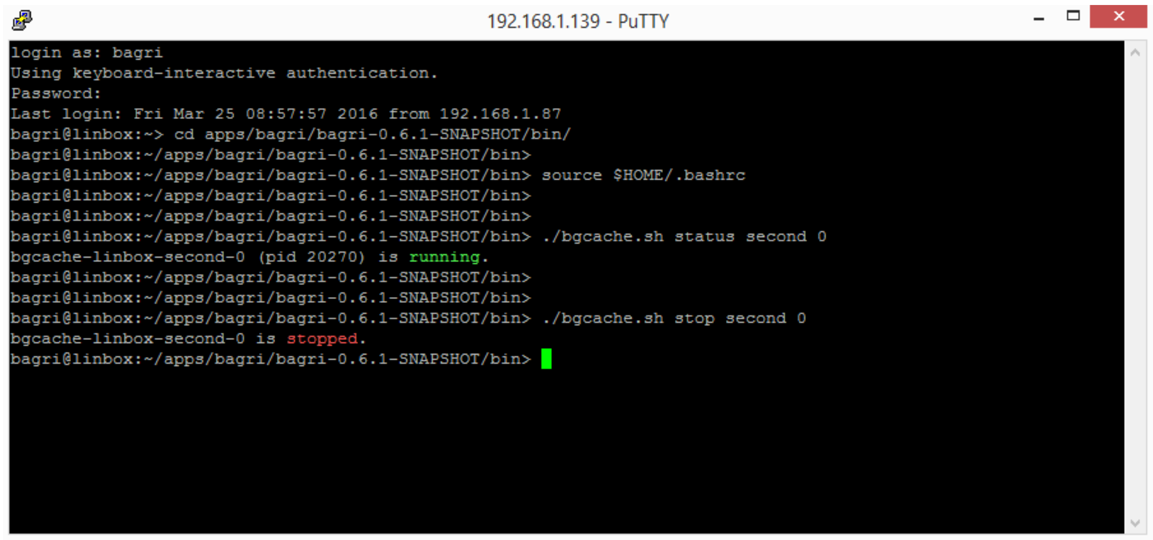
   `>./bgcache.sh status <profile-name> <node-number>`



```
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>
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

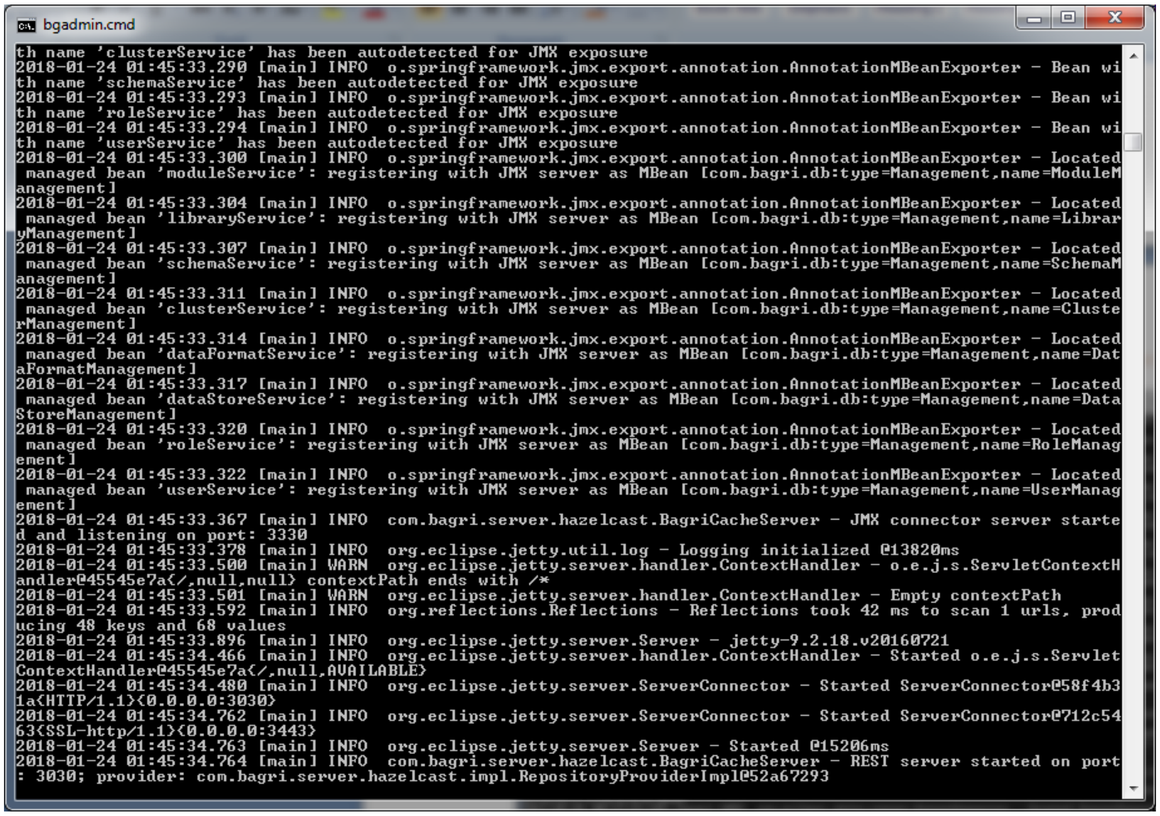
1. Navigate to **Start** → **Run** and type '"`cmd`". Type change directory command and path to the folder where `bgadmin.cmd`script is located and press **Enter**.

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

2. Type the following command:

   ```
   bgadmin.cmd
   ```

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



```
th name 'clusterService' has been autodetected for JMX exposure
2018-01-24 01:45:33.290 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Bean wi
th name 'schemaService' has been autodetected for JMX exposure
2018-01-24 01:45:33.293 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Bean wi
th name 'roleService' has been autodetected for JMX exposure
2018-01-24 01:45:33.294 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Bean wi
th name 'userService' has been autodetected for JMX exposure
2018-01-24 01:45:33.300 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'moduleService': registering with JMX server as MBean [com.bagri.db:type=Management,name=ModuleM
anagement]
2018-01-24 01:45:33.304 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'libraryService': registering with JMX server as MBean [com.bagri.db:type=Management,name=Librar
yManagement]
2018-01-24 01:45:33.307 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'schemaService': registering with JMX server as MBean [com.bagri.db:type=Management,name=SchemaM
anagement]
2018-01-24 01:45:33.311 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'clusterService': registering with JMX server as MBean [com.bagri.db:type=Management,name=Cluste
rManagement]
2018-01-24 01:45:33.314 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'dataFormatService': registering with JMX server as MBean [com.bagri.db:type=Management,name=Dat
aFormatManagement]
2018-01-24 01:45:33.317 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'dataStoreService': registering with JMX server as MBean [com.bagri.db:type=Management,name=Data
StoreManagement]
2018-01-24 01:45:33.320 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'roleService': registering with JMX server as MBean [com.bagri.db:type=Management,name=RoleManag
ement]
2018-01-24 01:45:33.322 [main] INFO  o.springframework.jmx.export.annotation.AnnotationMBeanExporter - Located
 managed bean 'userService': registering with JMX server as MBean [com.bagri.db:type=Management,name=UserManag
ement]
2018-01-24 01:45:33.367 [main] INFO  com.bagri.server.hazelcast.BagriCacheServer - JMX connector server starte
d and listening on port: 3330
2018-01-24 01:45:33.378 [main] INFO  org.eclipse.jetty.util.log - Logging initialized @13820ms
2018-01-24 01:45:33.500 [main] WARN  org.eclipse.jetty.server.handler.ContextHandler - o.e.j.s.ServletContextH
andler@45545e7a{/,null,null} contextPath ends with /*
2018-01-24 01:45:33.501 [main] WARN  org.eclipse.jetty.server.handler.ContextHandler - Empty contextPath
2018-01-24 01:45:33.592 [main] INFO  org.reflections.Reflections - Reflections took 42 ms to scan 1 urls, prod
ucing 48 keys and 68 values
2018-01-24 01:45:33.896 [main] INFO  org.eclipse.jetty.server.Server - jetty-9.2.18.v20160721
2018-01-24 01:45:34.466 [main] INFO  org.eclipse.jetty.server.handler.ContextHandler - Started o.e.j.s.Servlet
ContextHandler@45545e7a{/,null,AVAILABLE}
2018-01-24 01:45:34.480 [main] INFO  org.eclipse.jetty.server.ServerConnector - Started ServerConnector@58f4b3
1a{HTTP/1.1}{0.0.0.0:3030}
2018-01-24 01:45:34.762 [main] INFO  org.eclipse.jetty.server.ServerConnector - Started ServerConnector@712c54
63{SSL-http/1.1}{0.0.0.0:3443}
2018-01-24 01:45:34.763 [main] INFO  org.eclipse.jetty.server.Server - Started @15206ms
2018-01-24 01:45:34.764 [main] INFO  com.bagri.server.hazelcast.BagriCacheServer - REST server started on port
: 3030; provider: com.bagri.server.hazelcast.impl.RepositoryProviderImpl@52a67293
```

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

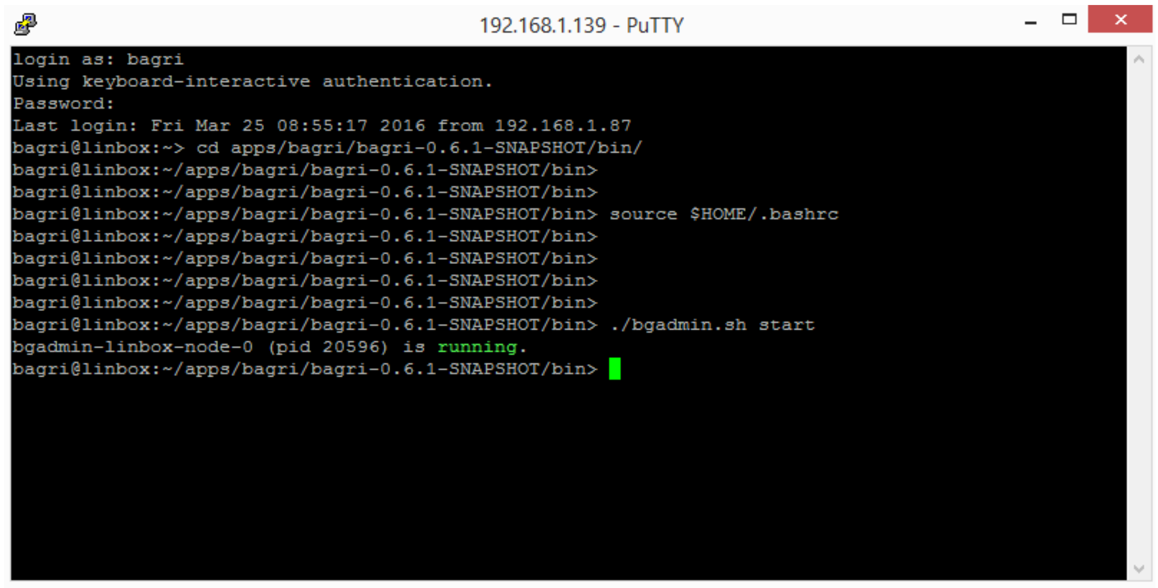
1. To run Administration server on a Linux OS you can use any terminal software such as PuTTY. Change directory to the `bin` folder where `bgadmin.sh` script is located and press **Enter**.

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

2. Input the following command:

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

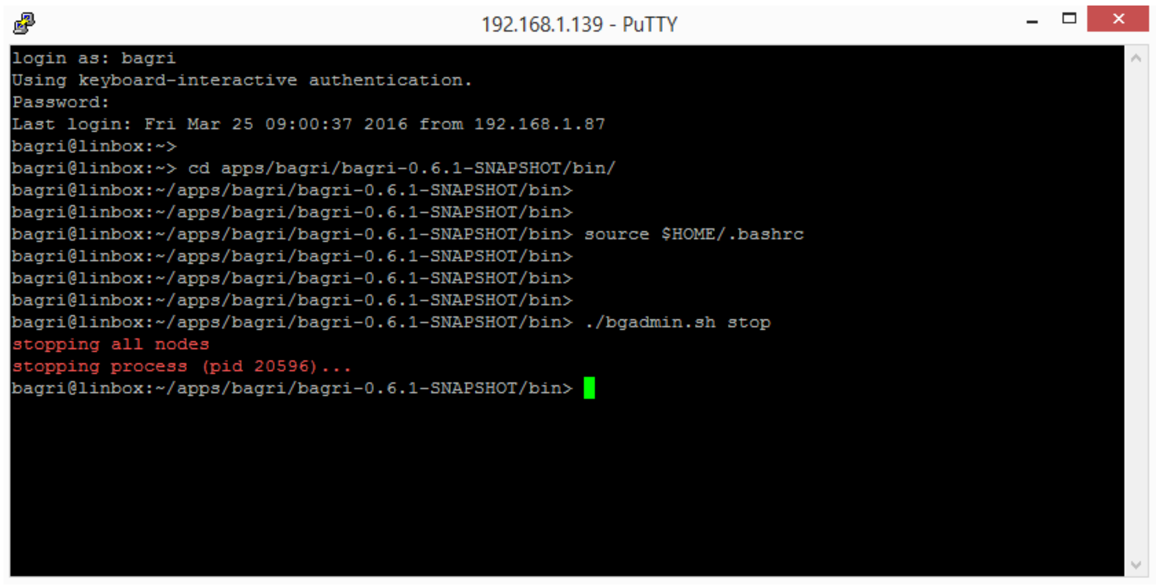
   On successful launch of the server, the script will display message that the node has been started properly:



```
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>
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>
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`



```
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>
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>
```

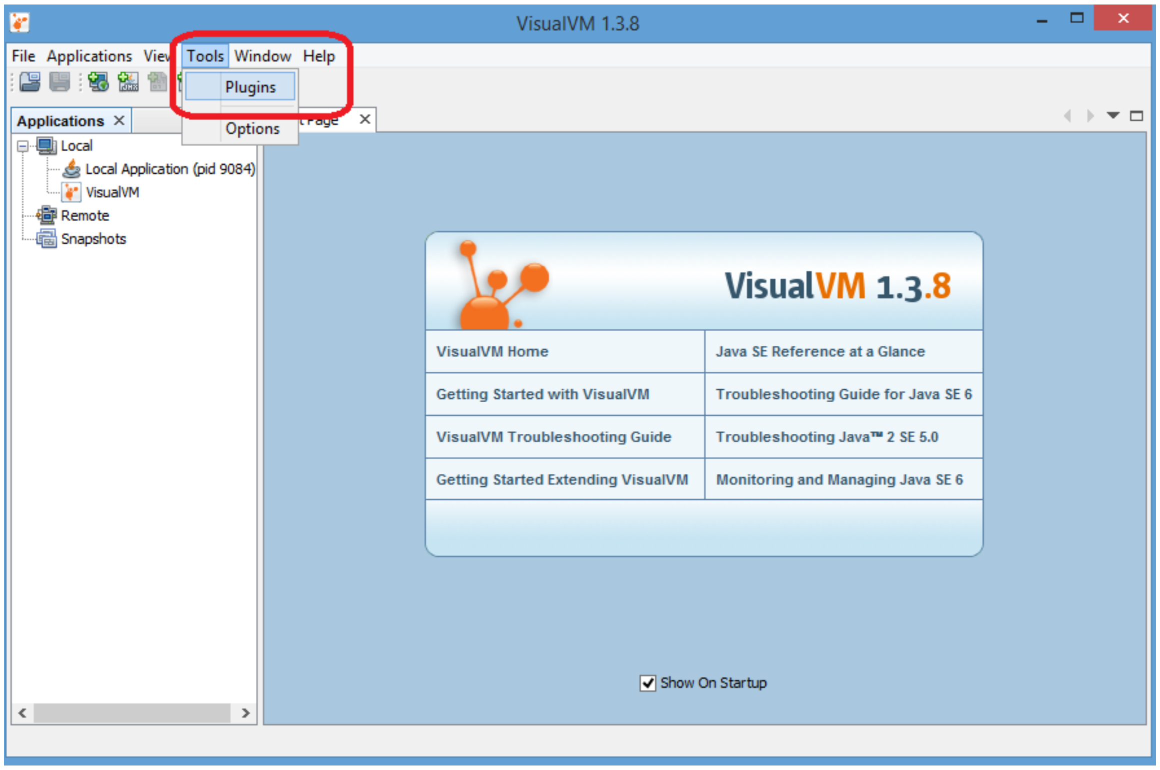
6. Configure Graphical User Interface (optional)

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.

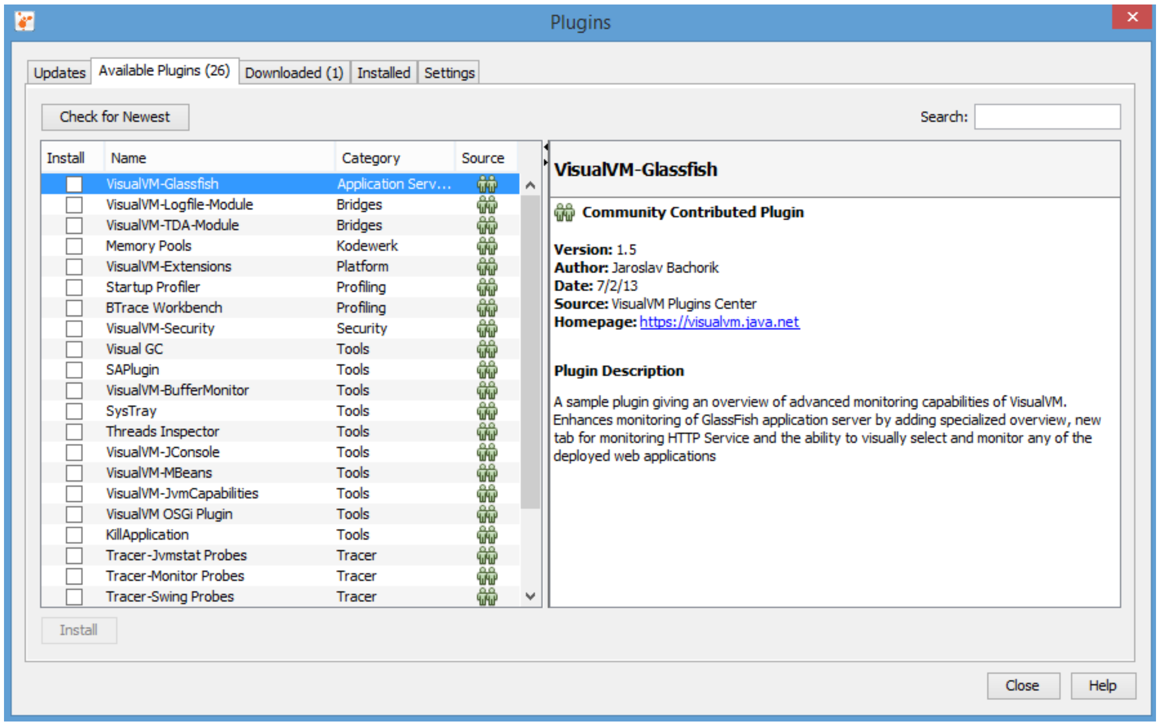
To Configure GUI on Windows platform

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.**


VisualVM 1.3.8
File
Applications
View
Tools
Window
Help
Plugins
Options
Applications
Local
Local Application (pid 9084)
VisualVM
Remote
Snapshots
VisualVM 1.3.8
VisualVM Home
Java SE Reference at a Glance
Getting Started with VisualVM
Troubleshooting Guide for Java SE 6
VisualVM Troubleshooting Guide
Troubleshooting Java™ 2 SE 5.0
Getting Started Extending VisualVM
Monitoring and Managing Java SE 6
Show On Startup

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



Plugins

Updates | Available Plugins (26) | Downloaded (1) | Installed | Settings

Check for Newest   Search:

Install	Name	Category	Source
☐	VisualVM-Glassfish	Application Serv...	
☐	VisualVM-Logfile-Module	Bridges	
☐	VisualVM-TDA-Module	Bridges	
☐	Memory Pools	Kodewerk	
☐	VisualVM-Extensions	Platform	
☐	Startup Profiler	Profiling	
☐	BTrace Workbench	Profiling	
☐	VisualVM-Security	Security	
☐	Visual GC	Tools	
☐	SAPlugin	Tools	
☐	VisualVM-BufferMonitor	Tools	
☐	SysTray	Tools	
☐	Threads Inspector	Tools	
☐	VisualVM-JConsole	Tools	
☐	VisualVM-MBeans	Tools	
☐	VisualVM-JvmCapabilities	Tools	
☐	VisualVM OSGi Plugin	Tools	
☐	KillApplication	Tools	
☐	Tracer-Jvmstat Probes	Tracer	
☐	Tracer-Monitor Probes	Tracer	
☐	Tracer-Swing Probes	Tracer	

VisualVM-Glassfish

Community Contributed Plugin

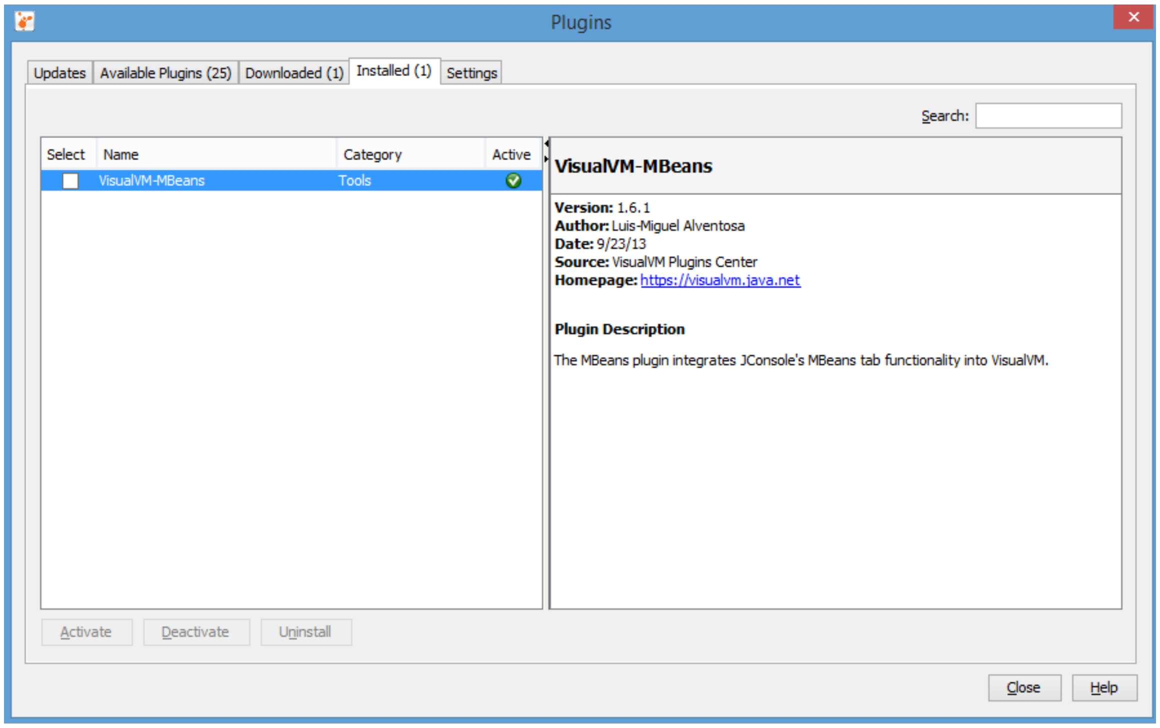
Version: 1.5
Author: Jaroslav Bachorik
Date: 7/2/13
Source: VisualVM Plugins Center
Homepage: https://visualvm.java.net

Plugin Description

A sample plugin giving an overview of advanced monitoring capabilities of VisualVM. Enhances monitoring of GlassFish application server by adding specialized overview, new tab for monitoring HTTP Service and the ability to visually select and monitor any of the deployed web applications

Install   Close   Help

- 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.



Plugins

Updates | Available Plugins (25) | Downloaded (1) | Installed (1) | Settings

Search:

Select	Name	Category	Active
☐	VisualVM-MBeans	Tools	✓

VisualVM-MBeans

Version: 1.6.1
Author: Luis-Miguel Alventosa
Date: 9/23/13
Source: VisualVM Plugins Center
Homepage: https://visualvm.java.net

Plugin Description

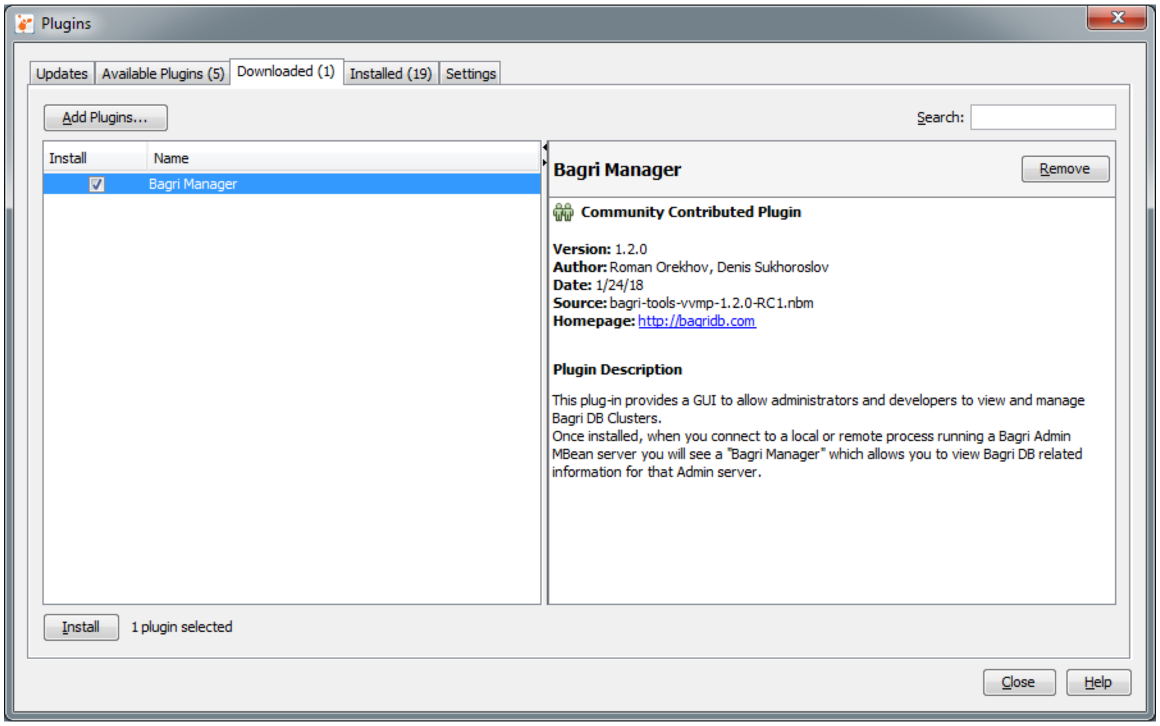
The MBeans plugin integrates JConsole's MBeans tab functionality into VisualVM.

Activate   Deactivate   Uninstall   Close   Help

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-1.2.1.nbm` and click **Open**.

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



☑ Bagri Manager

Bagri Manager

Community Contributed Plugin

Version: 1.2.0
Author: Roman Orekhov, Denis Sukhoroslov
Date: 1/24/18
Source: bagri-tools-vvmp-1.2.0-RC1.nbm
Homepage: http://bagridb.com

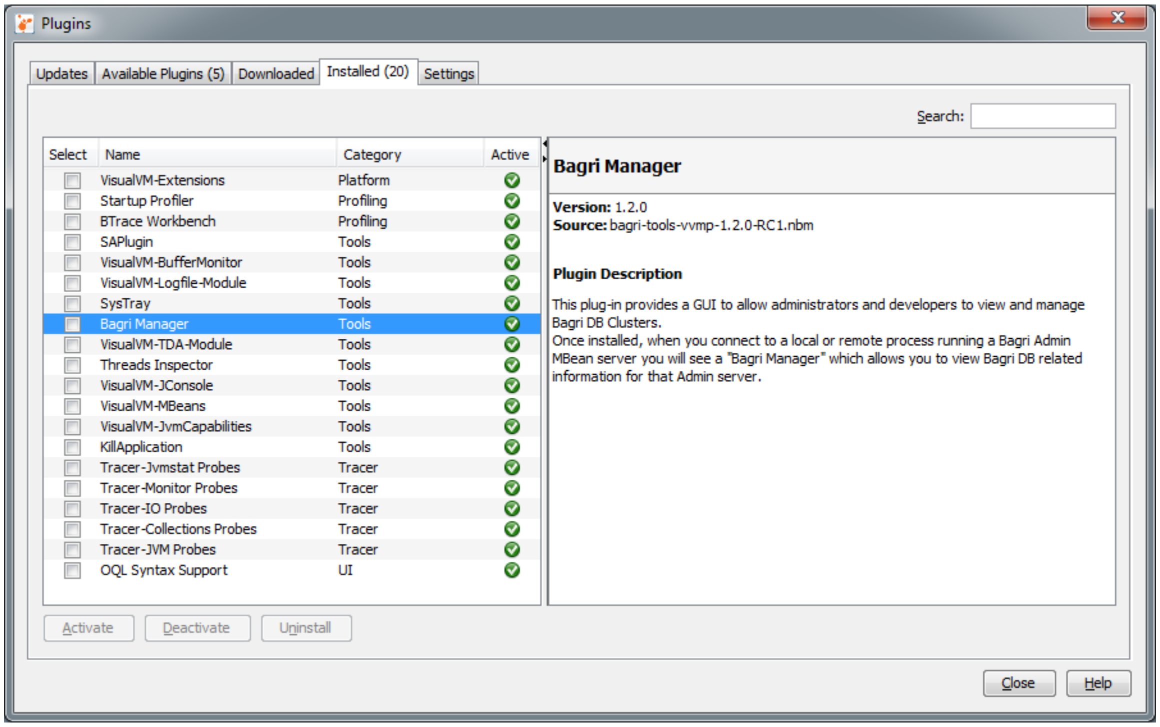
Plugin Description

This plug-in provides a GUI to allow administrators and developers to view and manage Bagri DB Clusters.
Once installed, when you connect to a local or remote process running a Bagri Admin MBean server you will see a "Bagri Manager" which allows you to view Bagri DB related information for that Admin server.

1 plugin selected

- 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:



Select	Name	Category	Active
☐	VisualVM-Extensions	Platform	✓
☐	Startup Profiler	Profiling	✓
☐	BTrace Workbench	Profiling	✓
☐	SAPlugin	Tools	✓
☐	VisualVM-BufferMonitor	Tools	✓
☐	VisualVM-Logfile-Module	Tools	✓
☐	SysTray	Tools	✓
☐	Bagri Manager	Tools	✓
☐	VisualVM-TDA-Module	Tools	✓
☐	Threads Inspector	Tools	✓
☐	VisualVM-JConsole	Tools	✓
☐	VisualVM-MBeans	Tools	✓
☐	VisualVM-JvmCapabilities	Tools	✓
☐	KillApplication	Tools	✓
☐	Tracer-Jvmstat Probes	Tracer	✓
☐	Tracer-Monitor Probes	Tracer	✓
☐	Tracer-IO Probes	Tracer	✓
☐	Tracer-Collections Probes	Tracer	✓
☐	Tracer-JVM Probes	Tracer	✓
☐	OQL Syntax Support	UI	✓

Bagri Manager

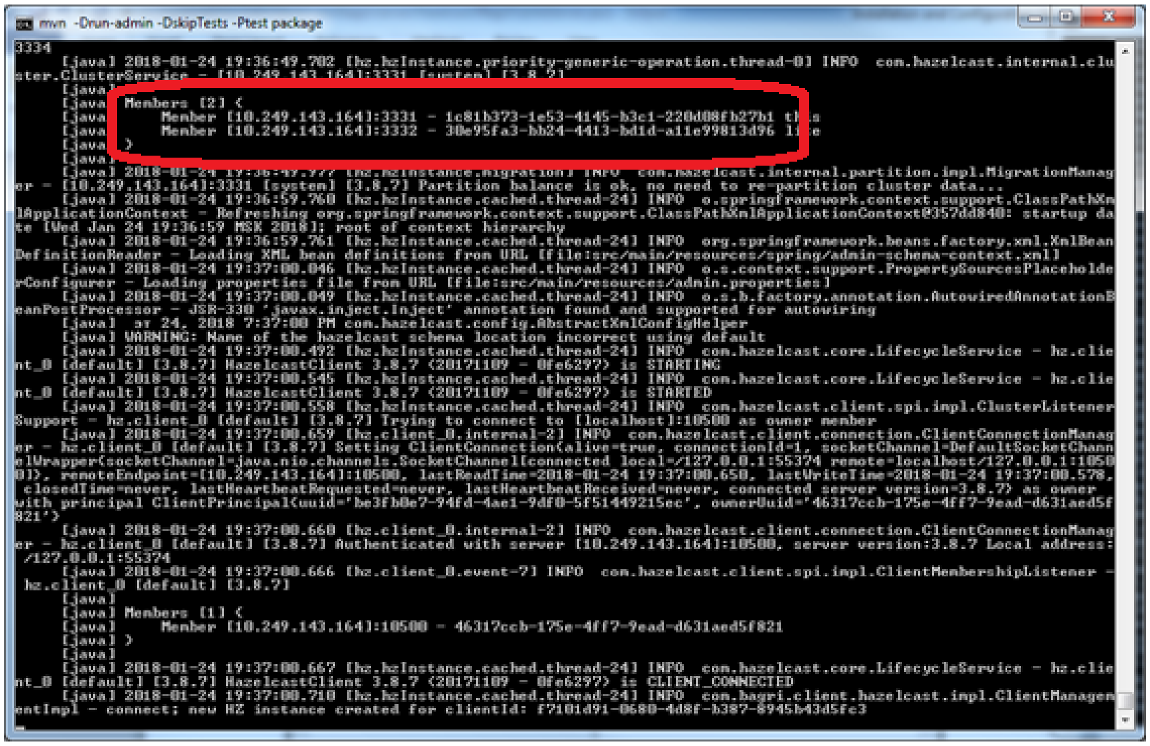
Version: 1.2.0
Source: bagri-tools-vvmp-1.2.0-RC1.nbm

Plugin Description

This plug-in provides a GUI to allow administrators and developers to view and manage Bagri DB Clusters.
Once installed, when you connect to a local or remote process running a Bagri Admin MBean server you will see a "Bagri Manager" which allows you to view Bagri DB related information for that Admin server.

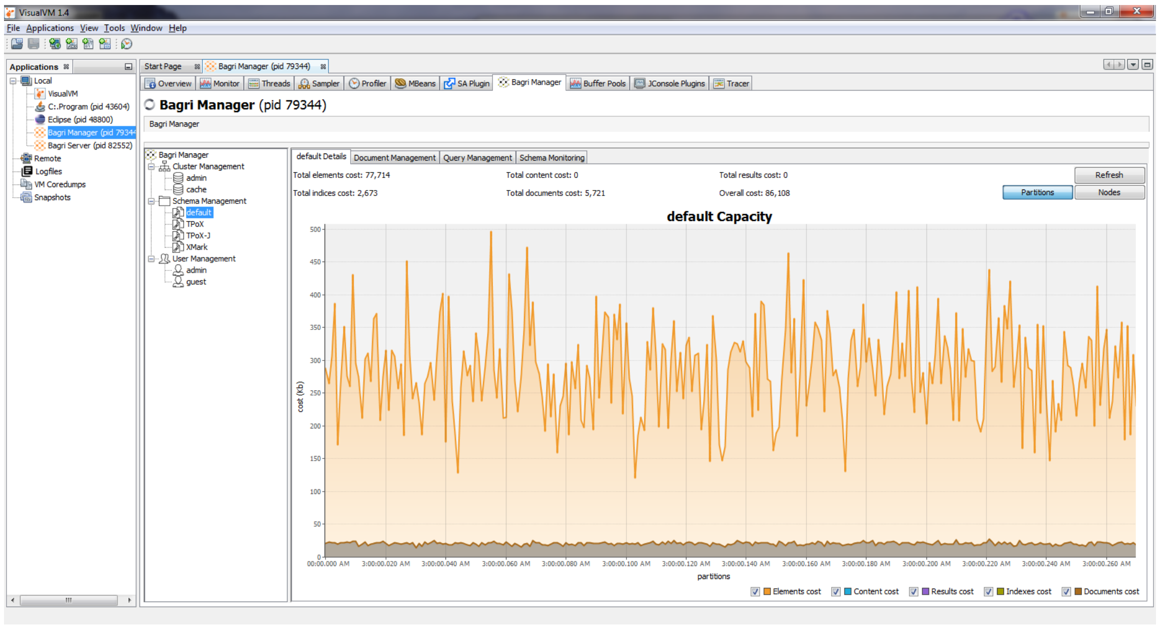
5. Open **Bagri Manager** plugin:

- Run Bagri Administrative and Data servers as it is explained above, ensure they're connected in their consoles or logs.



```
3334
     [java] 2018-01-24 19:36:49.702 [hz.hzInstance.priority-generic-operation.thread-0] INFO  com.hazelcast.internal.clu
ster.ClusterService - [10.249.143.164]:3331 [system] [3.8.7]
     [java
     [java  Members [2] {
     [java      Member [10.249.143.164]:3331 - 1c81b373-1e53-4145-b3c1-220d08fb27b1 th s
     [java      Member [10.249.143.164]:3332 - 30e95fa3-bb24-4413-bd1d-a11e99813d96 li e
     [java  }
     [java
     [java] 2018-01-24 19:36:49.977 [hz.hzInstance.migration] INFO  com.hazelcast.internal.partition.impl.MigrationManag
er - [10.249.143.164]:3331 [system] [3.8.7] Partition balance is ok, no need to re-partition cluster data...
     [java] 2018-01-24 19:36:59.760 [hz.hzInstance.cached.thread-24] INFO  o.springframework.context.support.ClassPathXm
lApplicationContext - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@357dd840: startup da
te [Wed Jan 24 19:36:59 MSK 2018]; root of context hierarchy
     [java] 2018-01-24 19:36:59.761 [hz.hzInstance.cached.thread-24] INFO  org.springframework.beans.factory.xml.XmlBean
DefinitionReader - Loading XML bean definitions from URL [file:src/main/resources/spring/admin-schema-context.xml]
     [java] 2018-01-24 19:37:00.046 [hz.hzInstance.cached.thread-24] INFO  o.s.context.support.PropertySourcesPlacehold e
rConfigurer - Loading properties file from URL [file:src/main/resources/admin.properties]
     [java] 2018-01-24 19:37:00.049 [hz.hzInstance.cached.thread-24] INFO  o.s.b.factory.annotation.AutowiredAnnotationB
eanPostProcessor - JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
     [java]  яг 24, 2018 7:37:00 PM com.hazelcast.config.AbstractXmlConfigHelper
     [java] WARNING: Name of the hazelcast schema location incorrect using default
     [java] 2018-01-24 19:37:00.492 [hz.hzInstance.cached.thread-24] INFO  com.hazelcast.core.LifecycleService - hz.clie
nt_0 [default] [3.8.7] HazelcastClient 3.8.7 (20171109 - 0fe6297) is STARTING
     [java] 2018-01-24 19:37:00.545 [hz.hzInstance.cached.thread-24] INFO  com.hazelcast.core.LifecycleService - hz.clie
nt_0 [default] [3.8.7] HazelcastClient 3.8.7 (20171109 - 0fe6297) is STARTED
     [java] 2018-01-24 19:37:00.558 [hz.hzInstance.cached.thread-24] INFO  com.hazelcast.client.spi.impl.ClusterListener
Support - hz.client_0 [default] [3.8.7] Trying to connect to [localhost]:10500 as owner member
     [java] 2018-01-24 19:37:00.659 [hz.client_0.internal-2] INFO  com.hazelcast.client.connection.ClientConnectionManag
er - hz.client_0 [default] [3.8.7] Setting ClientConnection{alive=true, connectionId=1, socketChannel=DefaultSocketChann
elWrapper{socketChannel=java.nio.channels.SocketChannel[connected local=/127.0.0.1:55374 remote=localhost/127.0.0.1:1050
0]}, remoteEndpoint=[10.249.143.164]:10500, lastReadTime=2018-01-24 19:37:00.650, lastWriteTime=2018-01-24 19:37:00.578,
 closedTime=never, lastHeartbeatRequested=never, lastHeartbeatReceived=never, connected server version=3.8.7} as owner
with principal ClientPrincipal{uuid='be3fb0e7-94fd-4ae1-9df0-5f5144921 5ec', ownerUuid='46317ccb-175e-4ff7-9ead-d631aed5f
821'}
     [java] 2018-01-24 19:37:00.660 [hz.client_0.internal-2] INFO  com.hazelcast.client.connection.ClientConnectionManag
er - hz.client_0 [default] [3.8.7] Authenticated with server [10.249.143.164]:10500, server version:3.8.7 Local address:
 /127.0.0.1:55374
     [java] 2018-01-24 19:37:00.666 [hz.client_0.event-7] INFO  com.hazelcast.client.spi.impl.ClientMembershipListener -
 hz.client_0 [default] [3.8.7]
     [java]
     [java] Members [1] {
     [java]     Member [10.249.143.164]:10500 - 46317ccb-175e-4ff7-9ead-d631aed5f821
     [java] }
     [java]
     [java] 2018-01-24 19:37:00.667 [hz.hzInstance.cached.thread-24] INFO  com.hazelcast.core.LifecycleService - hz.clie
nt_0 [default] [3.8.7] HazelcastClient 3.8.7 (20171109 - 0fe6297) is CLIENT_CONNECTED
     [java] 2018-01-24 19:37:00.710 [hz.hzInstance.cached.thread-24] INFO  com.bagri.client.hazelcast.impl.ClientManagem
entImpl - connect; new HZ instance created for clientId: f7101d91-0680-4d8f-b387-8945b43d5fc3
```

- Double click Bagri Manager node in the left hand side tree, then select Bagri Manager tab from the exlorer at the right hand side and choose schema which is started on your Data server(s). Then you can work with schema documents and perform (x)queries on them in the plugin tabs.



Total elements cost: 77,714
Total content cost: 0
Total results cost: 0
Total indices cost: 2,673
Total documents cost: 5,721
Overall cost: 86,108

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 templates for 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 conbines 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.
version	Account version
createdAt	Date and time when an account was created.
createdBy	The user who has created an account.

Property	Description
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>****************************</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 of options that can be used by cache servers. Each node has the following paramters:

Property	Description
name	The node template name
version	Node version
createdAt	Date and time when the template was created
createdBy	The user who has created the template
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

Property	Description
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 folowing 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. Posible 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 processing 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 formatplugin 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
handlerClass	The name of Java class implementing ContentHandler factory interface
extensions	File extensions which are registered with this data format
Type	The MIME type associated with data format
properties	An optional list of properties which will be passed to the handler class at initialization phase

The configsnipet below shows sample 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 Content Handler</description>
    <handlerClass>com.bagri.core.server.api.df.json.JsonpHandler</handlerClass>
    <enabled>true</enabled>
    <extensions>json</extensions>
    <type>application/json</type>
    <properties>
        <entry name="bdb.client.contentSerializer">
            com.bagri.client.hazelcast.serialize.StringContentSerializer
        </entry>
        <entry name="javax.json.spi.JsonProvider">
            org.glassfish.json.JsonProviderImpl
        </entry>
    </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 DB, etc. The data store plugin defined with the following parameters:

Property	Description
name	The data store name
version	Data storeplugin 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 storeat 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 folowing 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 schema

Property	Description
description	Textual definition of the schema
properties	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

Property	Description
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 paramters:

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

Property	Description
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`
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>
    <dataTypexmlns: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

Property	Description
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. There are document and transaction level triggers. The first ones are fired at the moment of creation, amendment or deletion of a document. The second ones applied at transaction begin/commit/rollback. 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
collection	If specified, the trigger will fire only on documents belonging to this collection
synchronous	Trigger can be fired synchronously (`true`) or asynchronously (`false`)
enabled	Trigger cab be enabled or not
actions	The list of action when trigger will be fired. Every action has the following attributes:
index	An order at which trigger will be fired

Property	Description
order	When the trigger will be fired, right `before` or `after`the `scope`
scope	A moment in document or transaction lifecycle when an event takes place. Possible values are: `insert/update/delete and begin/commit/rollback`
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>
    <actions>
        <action index="1" order="after" scope="insert"/>
        <action index="2" order="before" scope="commit"/>
    </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

Property name	Mandatory (Y/N)	Default or Sample Value	Description
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	relative path to teh folder containig schema document sources
bdb.schema.format.default		XML	the name of the data format plugin which will perform document's content processing
bdb.schema.population.size		1	schema population size (node count)
bdb.schema.population.buffer.size		1000000	schema catalog buffer size
bdb.schema.publish.counters		true	disables document counters publishing when set to false
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.connect.timeout		5	schema cluster connection timeout in seconds
bdb.schema.multicast.enabled		false	enables schema members dicovery via UDP
bdb.schema.tcp.enabled		true	enables schema discovery dicovery via TCP
bdb.schema.aws.enabled		false	enables schema dicovery in AWS cloud

Property name	Mandatory (Y/N)	Default or Sample Value	Description
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.level		readCommited	default transaction isolation level; valid values are: dirtyRead, readCommited, repeatableRead; empty value means no transaction
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.cache		true	enables query results caching
bdb.schema.query.parallel		true	to perform queries on partitions in parallel or not
bdb.schema.data.cache		NEVER	specifies data caching options; valid values are: NEVER, ALWAYS, INDEX-ONLY
bdb.schema.data.stats.enabled		true	enables collection of data statistics
bdb.schema.data.backup.async		1	number of schema documents asynchronous backups
bdb.schema.data.backup.sync		0	number of schema documents synchronous backups
bdb.schema.data.backup.read		false	allow to read schema documents from backup replica

Property name	Mandatory (Y/N)	Default or Sample Value	Description
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 override default plugin settings with its own values.

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

Property name	Mandatory (Y/N)	Default or Sample Value	Description
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 sequenecs
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.copyNamespacesModeInherit	N	1	Copy-namespaces inherit mode
xqj.schema.defaultOrderForEmptySequences	N	2	Default order for empty sequences
xqj.schema.defaultCollationUri	N	http://www.w3.org/2005/xpath-functions/collation/codepoint	Default collation
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.multicast.enabled	N	false	enables administrative cluster members discovery via UDP
bdb.cluster.tcp.enabled	N	true	enables administrative cluster members discovery via TCP
bdb.cluster.aws.enabled	N	false	enables administrative cluster members discovery in AWS cloud
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 clusterclient thread pool size

Property name	Mandatory (Y/N)	Default or Sample Value	Description
bdb.cluster.multicast.enabled	N	false	enables administrative cluster members discovery via UDP
bdb.cluster.tcp.enabled	N	false	enables administrative cluster members discovery via TCP
bdb.cluster.aws.enabled	N	false	enables administrative cluster members discovery in AWS cloud
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 in the profile.

8. Related Documentation

Document	Description

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