4. Cluster configuration

In order to run the Blueriq Runtime in a cluster environment, a number of configurations must be done:

- enable the Redis key-value store component
- configure Security Context Repository
- configure Session Manager
- configure HTTP Session Store
  - Session Timeout
  - Session Cookies

Each of these steps are detailed in the following sections.

Enable Redis Key-Value Store

By default, Blueriq provides a Key-Value Store Component that works with Redis. For details on how to configure and enable it, please check the Key-Value Store documentation.

Some sections described in this chapter depend on the Key-Value Store thus an enabled key-value store component must be present in the runtime before continuing

Configure Session Manager

The session manager is responsible for storing IAquimaSession instances. When running in a cluster, IAquimaSessions must also be stored in a key-value store. In order to configure the Runtime to store IAquimaSessions in a key-value store the following setting must be made in application.properties:

```
blueriq.session.session-manager=external
```

For more information about the Session manager, visit the Blueriq Session Manager documentation page.

Configure HTTP Session Store

By default, HTTP sessions are stored in memory and sessions are managed by the application container. When running in a cluster, the HTTP sessions must be stored in an external session store. Blueriq leverages Spring Boot and Spring Session to configure the HTTP session store.

HTTP Session Store Settings

Blueriq uses Spring Boot and Spring Session to set where HTTP sessions are stored and which component manages HTTP sessions. The `spring.session.store-type` property can be used to select the session store. The following implementations are supported by default:

Store Type	Managed By	Description
none	Application Container	The default value. Sessions are stored in memory and are managed by the application container.
redis	Spring Session	Sessions are stored in Redis
mongo	Spring Session	Sessions are stored in MongoDB
JDBC	Spring Session	Sessions are stored in a relational database
hazelcast	Spring Session	Sessions are stored in Hazelcast
hash_map	Spring Session	Sessions are stored in memory. This implementation is intended for testing and should not be used in production.

Blueriq provides support for storing HTTP sessions in a single Redis instance. Storing sessions in a Redis cluster or in other store types is not yet supported.

The connection to the external session stores is configured using standard Spring Boot properties. For example, configuring the connection to Redis can be done with the following properties:

```
spring.data.redis.host=redis.example.com
spring.data.redis.password=example
spring.data.redis.port=6379
```

Information on how to configure the redis connection pool can be found here.

Session Timeout Settings

The session timeout is configured differently depending on where the session is stored and whether the Runtime is deployed in an application container or is running in standalone mode.

Deployment	Session Store	Session Timeout Configuration
Application Container	none	The session timeout can be configured from the application container.
Application Container	redis	The `server.servlet.session.timeout` property can be used to set the session timeout in seconds.
Standalone	none	
Standalone	redis	

Session Cookie Settings

The session cookie is configured differently depending on where the session is stored and whether the Runtime is deployed in an application container or is running in standalone mode:

Deployment	Session Store	Session Cookie Configuration
Application Container	none	The session cookie is configured from the application container. Please note that the Runtime always overrides the HttpOnly and Secure flags. See Security: Blueriq session & cookie

Application Container	redis	The standard Spring Boot properties can be used:
		`server.servlet.session.cookie.name=JSESSIONID`
		`server.servlet.session.cookie.http-only=true`
		`server.servlet.session.cookie.secure=true`
		`server.servlet.session.cookie.path=/Runtime`

Standalone	none	
Standalone	redis	

HTTP Session Listeners

If Redis is used as HTTP session store and the application uses HTTP Session listeners (from Servlet API, Spring Security or Spring Session), Keyspace Notifications must be enabled in Redis by running the following command in the Redis client:

```
config set notify-keyspace-events "g$xKE"
```

Additionally, Servlet API HttpSessionListeners must be exposed as Spring Components.

Example Configuration

In order to store the security context, IAquimaSessions and HTTP sessions in Redis, the following configuration can be used:

1. Enable the Redis Key-Value Store Component in bootstrap.properties

```
spring.profiles.active=native,keyvalue-redis-store
```

2. Set the HTTP session store type to `redis` and configure the Redis connection for the HTTP session store in application.properties (the same Redis instance used by the Redis Key-Value Store Component can be used):

```
spring.data.redis.host=redis.example.com
spring.data.redis.password=example
spring.data.redis.port=6379

spring.session.store-type=redis
```

3. Configure the session timeout and session cookie

```
server.servlet.session.timeout=300

server.servlet.session.cookie.name=JSESSIONID
server.servlet.session.cookie.http-only=true
server.servlet.session.cookie.secure=true
server.servlet.session.cookie.path=/Runtime
```

4.Configure the Session manager for storing data in the Key Value Store

```
blueriq.session.session-manager=external
```

Previous: 3. Key-value store API and default component

Next: 5. Blueriq Session Manager