# Exercise 9: AMQ 7.12 Message Persistence & Replication

AMQ7.12 has various message persistence options:

* NIO (Java NIO, journal-based persistence)
* ASYNCIO (Linux Asynchronous IO, journal-based persistence)
* JDBC (persist to the relational database of your choice)
* Memory (in-memory stateless message persistence)

The default, out-of-the-box settin is NIO file journal-based persistence. You can configure message persistence by updating the /jboss-amq-7.0.0.redhat-1/brokers/myfirstbroker/etc/broker.xml file. Here is the section in broker.xml:

<configuration>

<core>

...

<persistence-enabled>true</persistence-enabled>

<!-- this could be ASYNCIO or NIO-->

<journal-type>NIO</journal-type>

<paging-directory>./data/paging</paging-directory>

<bindings-directory>./data/bindings</bindings-directory>

<journal-directory>./data/journal</journal-directory>

<large-messages-directory>./data/large-messages</large-messages-directory>

<journal-datasync>true</journal-datasync>

<journal-min-files>2</journal-min-files>

<journal-pool-files>-1</journal-pool-files>

...

</core>

</configuration>

## Replication

The following lab demonstrates setting up a 2-node master / slave cluster with shared-nothing replication.

### Lab - Shared-nothing replication between Master / Slave cluster

#### Prerequisites

Download the legacy ActiveMQ client JAR which supports OpenWire failover [here](https://mvnrepository.com/artifact/org.apache.activemq/activemq-all/5.14.5)

1. Create a master and slave broker pair by running the following commands:

$ ./bin/artemis create brokers/master

$ ./bin/artemis create brokers/slave --port-offset 1

Be sure to give each broker the admin/admin username / password combo and allow anonymous access.

1. Replace the etc/broker.xml file from the examples directory for both master and slave:

cp /jboss-amq-7.0.0.redhat-1/examples/features/clustered/clustered-static-discovery/src/main/resources/activemq/server0/broker.xml /brokers/master/etc/broker.xml

cp /jboss-amq-7.0.0.redhat-1/examples/features/clustered/clustered-static-discovery/src/main/resources/activemq/server1/broker.xml /brokers/slave/etc/broker.xml

1. Replace the security-settings section of both master / slave broker.xml files with the following text (this will allow the producer / consumer to dynamically create queues):

<security-settings>

<security-setting match="#">

<permission type="createNonDurableQueue" roles="amq"/>

<permission type="deleteNonDurableQueue" roles="amq"/>

<permission type="createDurableQueue" roles="amq"/>

<permission type="deleteDurableQueue" roles="amq"/>

<permission type="createAddress" roles="amq"/>

<permission type="deleteAddress" roles="amq"/>

<permission type="consume" roles="amq"/>

<permission type="browse" roles="amq"/>

<permission type="send" roles="amq"/>

<!-- we need this otherwise ./artemis data imp wouldn't work -->

<permission type="manage" roles="amq"/>

</security-setting>

</security-settings>

1. Update the cluster-connection section by replacing the message-load-balacing line with the following:

<message-load-balancing>ON\_DEMAND</message-load-balancing>

This will prevent message starvation and enable message redistribution between nodes.

1. Startup both the master and slave brokers in separate consoles

./brokers/master/bin/artemis run

./brokers/slave/bin/artemis run

1. Using the legacy activemq client, run the following commands in two separate console windows:

java -jar activemq-all-5.11.0.redhat-630187.jar consumer --brokerUrl 'failover:(tcp://localhost:61616,tcp://localhost:61617)' --user admin --password admin --destination queue://TEST

java -jar activemq-all-5.11.0.redhat-630187.jar producer --sleep 100 --messageCount 1000 --user admin --password admin --brokerUrl 'failover:(tcp://localhost:61616,tcp://localhost:61617)' —destination queue://TEST

1. Kill the master broker, and observe failover of both consumer / producer processes to the slave broker
2. Startup the original master broker again. Kill the slave broker, and notice failover back to the original master.