

High Availability and Failover Configurations of ActiveMQ Artemis

Introduction

High availability is when the system can keep working even if one or more servers stop working. Failover is a part of high availability and it means that if one server (live server) fails, the client connections can move to another server (backup server) so that the client applications can keep working without any interruption.

Apache ActiveMQ Artemis supports two different strategies for backing up a server *shared store* and *replication*. Which is configured via the ha-policy configuration element.

- **Replication**
Replicated clusters in ActiveMQ Artemis are setups where the data is synchronized between the master and slave brokers. When messages are sent to the master broker, the data is replicated to all the configured slave brokers. This ensures that even if the master broker goes down, the data is available on the slave brokers, allowing them to take over the role of the master and continue serving clients.
- **Shared-store**
When using a shared store, both live and backup servers share the same entire data directory using a shared file system. When failover occurs and a backup server takes over, it will load the persistent storage from the shared file system and clients can connect to it.

Both servers we use replicated clustered configurations.

Installation ActiveMQ Artemis

Before install Artemis make sure you have Java 11 or later version installed on your machine. If not you can install JDK from the following link:
<https://www.oracle.com/java/technologies/downloads/>

In this project, I used two different operating systems. It is not a necessity for them to be different, but I personally use MacOS and set up a Linux virtual machine through Parallels. For this reason, I will provide installation details for both operating systems. If you also want to install Parallels for Mac (M1), you can do so through this website: <https://www.parallels.com/eu/>

Install ActiveMQ on MacOS (M1)

- **Download Apache ActiveMQ Artemis:** Visit the official Apache ActiveMQ Artemis website (<https://activemq.apache.org/components/artemis/download/>) and download the latest version of the software.

- **Extract the Archive:** Once the download is complete, extract the downloaded archive to a location of your choice on your Mac.
- **Set Environment Variables:** Open the Terminal and set the *ARTEMIS_HOME* environment variable to point to the extracted folder's location. For example:

```
$ export ARTEMIS_HOME=/path/to/extracted/folder
```

- **Create a Broker Instance:** Run the following command to create a new broker instance:

```
$ cd $ARTEMIS_HOME/bin
$ ./artemis create <instance-name>
```

```
nagkim@Nagkichans-MacBook-Air bin % ./artemis create test
Creating ActiveMQ Artemis instance at: /Users/nagkim/Downloads/apache-artemis-2.
29.0/bin/test

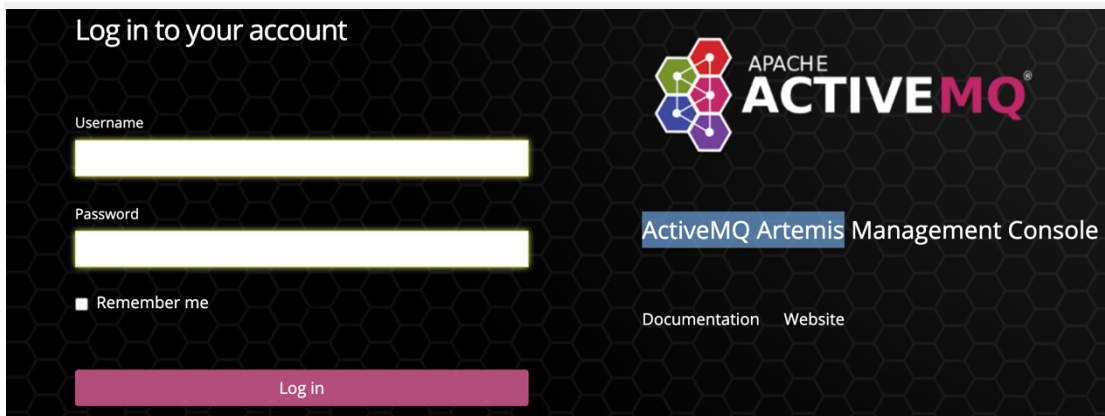
--user:
What is the default username?
admin

--password: is mandatory with this configuration:
What is the default password?

--allow-anonymous | --require-login:
Allow anonymous access?, valid values are Y, N, True, False
Y

Auto tuning journal ...
done! Your system can make 0.25 writes per millisecond, your journal-buffer-time
out will be 3995999
```

- **Start the Server:** After creating the broker instance, navigate to its bin directory and start the server:



Log in to your account

Username

Password

☐ Remember me

Log in

APACHE
ACTIVEMQ

ActiveMQ Artemis Management Console

[Documentation](#) [Website](#)

Install ActiveMQ on Ubuntu 18.04

- We should not run Artemis under the root user for security reasons. So, we need to create a group artemis and add a user artemis and we are going to install Artemis under */opt/artemis* directory:

```
$ sudo groupadd artemis
$ sudo useradd -s /bin/false -g artemis -d /opt/artemis artemis
```

- Now we can download Artemis version 2.29.0 on the */opt* directory

```
$ cd /opt
$ sudo wget https://archive.apache.org/dist/activemq/activemq-artemis/2.29.0/apache-artemis-2.29.0-bin.tar.gz
```

Version of slave device should be same version as master. My master's version is 2.29.0.

- After finish the download extract the tar package and rename the extracted directory to **artemis**

```
$ sudo tar -xvzf apache-artemis-2.8.1-bin.tar.gz
$ sudo mv apache-artemis-2.8.1 artemis
```

- We need to change the permission and the ownership of the Artemis Home directory. We will also give executed permission to *opt/artemis/bin/* directory.

```
$ sudo chown -R artemis: artemis
$ sudo chmod o+x /opt/artemis/bin/
```

- And finally we can create an instance like we do in the Mac. I will create into */var/lib* directory.

```
$ cd /var/lib
$ sudo /opt/artemis/bin/artemis create test-broker
```

```

parallels@ubuntu-linux-22-04-desktop:/$ sudo /opt/artemis/bin/artemis create test-broker
[sudo] password for parallels:
Creating ActiveMQ Artemis instance at: /test-broker

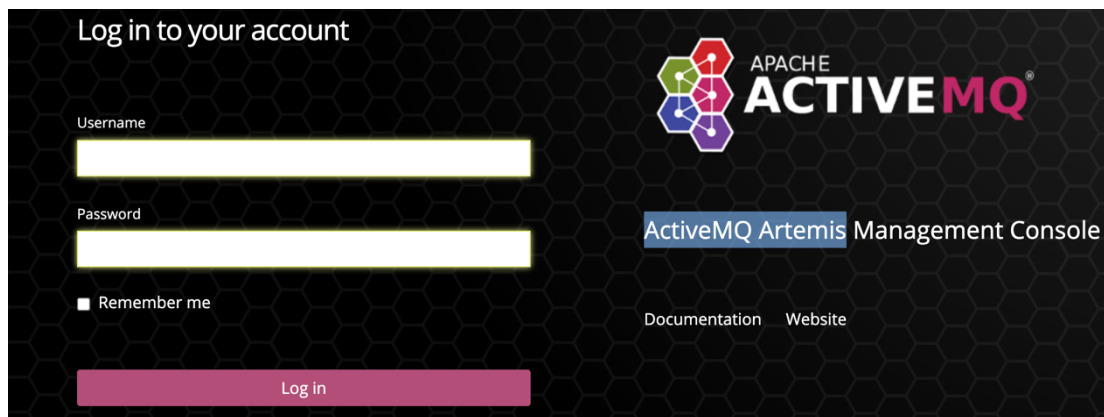
--user:
What is the default username?
admin
--setting match="*"
--password: is mandatory with this configuration:
What is the default password? roles="amq" />
session type="createDurableQueue" roles="amq"/>
session type="deleteDurableQueue" roles="amq"/>
--allow-anonymous | --require-login:
Allow anonymous access?, valid values are Y, N, True, False
N
session type="consume" roles="amq"/>
session type="browse" roles="amq"/>
session type="send" roles="amq"/>
Auto tuning journal ... (emis data imp wouldn't work ...)
done! Your system can make 19.23 writes per millisecond, your journal-buffer-timeout will be 52000
--setting-
You can now start the broker by executing:
"/test-broker/bin/artemis" run
you can also run the broker in the background using:
--setting match="activemq.management*"
Or you can run the broker in the background using:
--address=ExpiryQueue>/expiry-address>
--setting-
"/test-broker/bin/artemis-service" start

```

- We can run using the below command

```
$ sudo /var/lib/test-broker/bin/artemis run
```

- Once the broker instance starts we can access the admin console at <http://localhost:8161/console> and the login page will appear.



Log in to your account

Username

Password

☐ Remember me

[Documentation](#) [Website](#)

[Log in](#)

APACHE
ACTIVEMQ[®]

ActiveMQ Artemis Management Console

Create Replicated Brokers

After setting up the installations and test brokers for the two different operating systems, we can now proceed to configure the replica cluster configurations. To make servers replicated we can create as replicated clustered. On master device we need to create master instance like the below command:

```
$ ./artemis create --replicated --clustered master
```

For slave broker we need to run slave command and create the backup instance on Ubuntu.

```
$ ./artemis create --replicated --clustered slave
```

After we run these commands both of the systems ask you for additional inputs:

- **--user** : it is the default username to login Artemis.
- **--password** : it is the login password.
- **--host** : your device IP (master's or slave's).
- **--cluster-user** : a username for cluster connection.
- **--cluster-password** : password for cluster connection.
- **--allow-anonymous** | **--require-login** : allow anonymous access or not.

```
nagkim@Nagkichans-MacBook-Air bin % ./artemis create --replicated --clustered master1
Creating ActiveMQ Artemis instance at: /Users/nagkim/Downloads/apache-artemis-2.29.0/bin/master1

--user:
What is the default username?
admin

--password: is mandatory with this configuration:
What is the default password?

--host:
Host 0.0.0.0 is not valid for clustering. Please provide a valid IP or hostname
localhost

--cluster-user:
What is the cluster user?
admin

--cluster-password: is mandatory with this configuration:
What is the cluster password?

--allow-anonymous | --require-login:
Allow anonymous access?, valid values are Y, N, True, False
N

Auto tuning journal ...
done! Your system can make 0.25 writes per millisecond, your journal-buffer-timeout will be 4008000
```

However, some manual changes may still be required to set up the replication. We need to edit *broker.xml* files to each instances as I will explained below.

Cluster Configurations Of Master And Slave Brokers

Artemis provides cluster configuration to create a group of brokers that work together. This cluster setup allows multiple brokers to share the workload and handle failover seamlessly. In case one broker fails, another can take over to ensure uninterrupted message processing. Both brokers include these three main components:

I. `<broadcast-groups>` :

- Helps brokers discover each other even if they are on different network segments

```
<broadcast-groups>
  <broadcast-group name="gss-broadcast">
    <group-address>${udp-address:231.7.7.7}</group-address>
    <group-port>9876</group-port>
    <broadcast-period>100</broadcast-period>
    <connector-ref>artemis</connector-ref>
  </broadcast-group>
</broadcast-groups>
```

II. `<discovery-groups>` :

- Allows brokers to learn about each other dynamically and join the cluster.

```
<discovery-groups>
  <discovery-group name="gss-discovery">
    <group-address>${udp-address:231.7.7.7}</group-address>
    <group-port>9876</group-port>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

III. `<cluster-connections>` :

- Enables brokers to share information, such as message state and data replication, to achieve high availability and failover. Slave cluster can discover if a live server is already running, see *check-for-live-server*.
- Master broker: include `<use-duplicate-detection>` for filtering out duplicate messages without you having to code your own fiddly duplicate detection logic at the application level.

```
<cluster-connections>
  <cluster-connection name="gss-cluster">
    <address>jms</address>
    <connector-ref>artemis</connector-ref>
    <retry-interval>500</retry-interval>
    <use-duplicate-detection>true</use-duplicate-detection>
    <message-load-balancing>ON_DEMAND</message-load-balancing>
    <max-hops>1</max-hops>
    <static-connectors>
      <connector-ref>artemis</connector-ref>
    </static-connectors>
  </cluster-connection>
</cluster-connections>
```

- Slave broker : same configuration but without duplicate detection.
- To connect clusters we need to add user and password information

```
<cluster-user>admin</cluster-user>
<cluster-password>admin</cluster-password>
```

We should make the master and slave configurations to allow to be linked together as *live - backup* groups.

Configuration Of Master Cluster

Master cluster is the live server until it is down... After we are done with cluster and groups we need to add HA Policies.

This is the HA policies of master:

```
<ha-policy>
  <replication>
    <master>
      <check-for-live-server>true</check-for-live-server>
    </master>
  </replication>
</ha-policy>
```

<check-for-live-server> : This option is only necessary for performing 'fail-back' on replicating servers. If set, backup servers will only pair with live servers with matching group-name.

NOTE:

We need to start with **<connectors>** and **<acceptors>**. For master cluster we need to change the connection and also acceptors IP's with master device IP.

```
<connectors>
  <connector name="artemis">tcp://Master_device_IP:61616</connector>
</connectors>
```

```
<acceptor name= "artemis">tcp://Master_device_IP:61616?tcpSendBufferSize=
1048576;tcpReceiveBufferSize=1048576;amqpMinLargeMessageSize=102400;protocol
s=CORE,AMQP,STOMP,HORNETQ,MQTT,OPENWIRE;useEpoll=true;amqpCredits=1000;amqpL
rnlManagementObjects=false
</acceptor>
```


Configuration Of Slave Cluster

Slave broker is the backup server that when live server is down the backup server will replace and become the current live server. Here are some HA policies for the slave broker that we need to make:

- **<allow-failback>** : Whether a server will automatically stop when a another places a request to take over its place. The use case is when the backup has failed over

```
<ha-policy>
  <replication>
    <slave>
      <allow-failback>true</allow-failback>
    </slave>
  </replication>
</ha-policy>
```

- **<connectors>** : These connectors allow clients to connect to both the master and slave brokers in our messaging system.

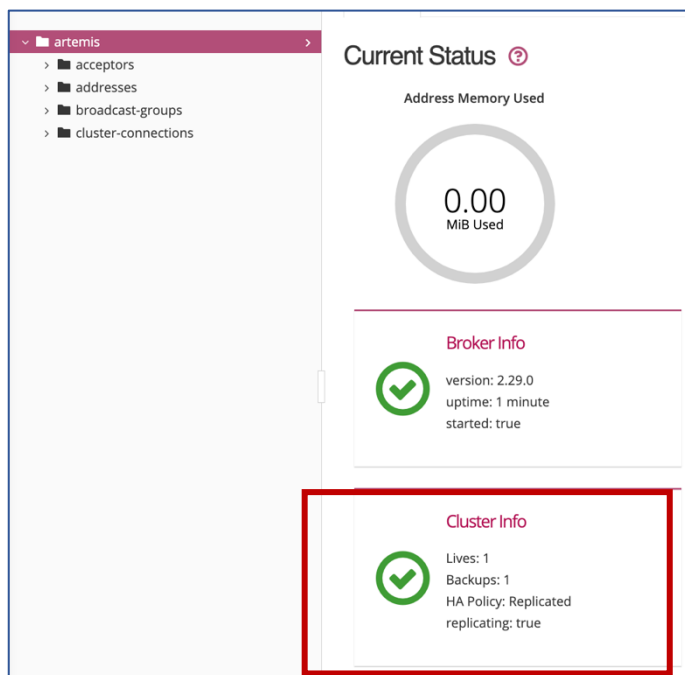
```
<connectors>
  <connector name="master-broker">tcp://Master_device_IP:61616</connector>
  <connector name="slave-broker">tcp://Slave_device_IP:61616</connector>
</connectors>
```

Running Servers

After we done with configurations we can test by running each broker.

- First run the master broker on Mac and open the login page <http://localhost:8161/console> from your browser.
- Now we can run the slave broker on Ubuntu and open the login page.
- We can see from Cluster Info section that backup server is connected to the master server.
- If we kill the master the slave automatically will be the current live server (but like copy of master).

❖ Master broker before kill it.



❖ After we kill master the slave become live server.

