



integrata  
cegos

# Apache Artemis

Eine Messaging System

- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
  - LPGL Lizenzmodell
- Schwerpunkt des Seminars ist die Administration
  - Grundlagen des Betriebs von Java Applikationsservern und ein grundlegendes Verständnis von Messaging-Systemen ist als Vorkenntnis zu empfehlen
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
  - <https://activemq.apache.org/components/artemis/>



[Home](#) [Components](#) [Contact](#) [Apache](#)

# ActiveMQ Artemis

The Next Generation Message Broker by ActiveMQ

Download



Read the Docs



APACHE  
**ACTIVEMQ**<sup>TM</sup>

## Power Your Microservices

### High-performance Messaging for Highly Scalable Microservices

Microservices are often implemented with HTTP using a "blocking" request-response pattern. This pattern can yield good latency in low throughput use-cases. However, patterns based on eventing and asynchronous messaging can deliver superior scalability and overall lower latency in high throughput use-cases. With potential throughput measured in the *millions* of messages per second, ActiveMQ Artemis has the performance and feature-set to bring these gains to your applications.

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6

81373 München

**Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.**

Übersicht	6
Konfiguration	21
Übersicht des Programmiermodells	37
Monitoring im Detail	46
System-Architekturen	58

1

# ÜBERSICHT

1.1

# INSTALLATION

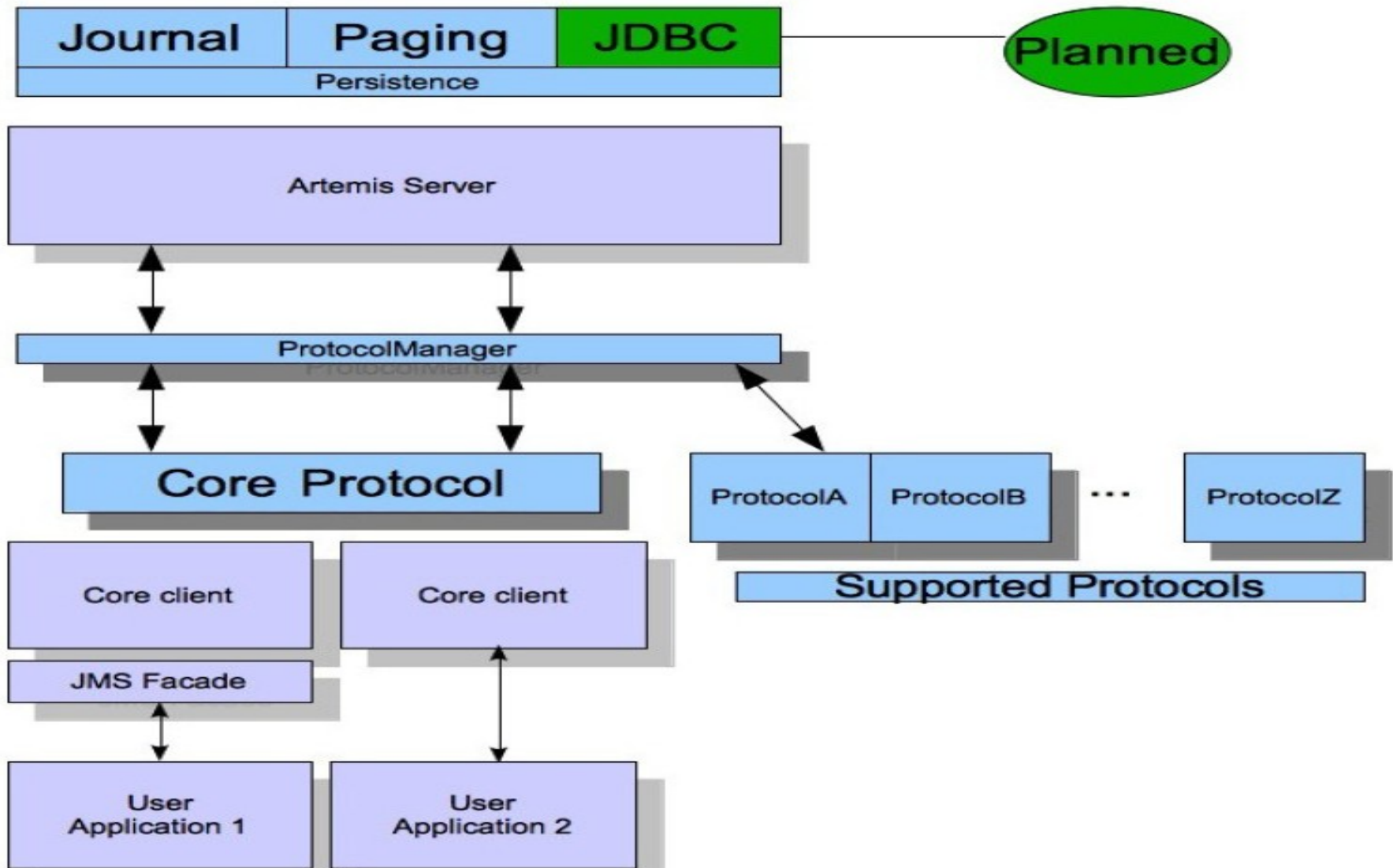
- Verteilung der Artemis-Distribution als Archiv
- Bisher kein offizielles Docker-Image auf dem Docker Hub vorhanden
- Allerdings existieren auf dem offiziellen Docker-Account des Artemis-Teams `Dockerfiles`, die die typische Installationsroutine beschreiben
  - <https://github.com/apache/activemq-artemis/tree/master/artemis-docker>



1.2

## **ARCHITEKTUR**

# Artemis Gesamtarchitektur (Quelle: Artemis Dokumentation)



1.3

## **DER ARTEMIS SERVER**

- Die nach dem Entpacken des Distributions-Archivs vorhanden Installation ist nicht ein der Artemis-Server (!)
  - Hilfsprogramm zur Konsolen-basierten Verwaltung und Benutzung eines laufenden Servers
  - Server-Generator

```
./artemis help
```

```
usage: artemis <command> [<args>]
```

The most commonly used artemis commands are:


address	Address tools group
browser	It will browse messages on an instance
consumer	consume messages from an instance
create	creates a new broker instance
data	data tools group (print)
help	Display help information
mask	mask a password and print it out
producer	It will send messages to an instance
queue	Queue tools group

```
./artemis help create
```

```
//Große Menge von Optionen, die allen möglichen  
Konfigurationseinstellungen eines Server entsprechen
```

- `./artemis create ~/artemis_training/first`
  - Erzeugt einen Standalone-Broker mit Standard-Einstellungen
- Verzeichnisstruktur
  - `bin`
    - Skripte zum Ausführen des Brokers
  - `data`
    - Persistent Storage für Messages
  - `etc`
    - Konfigurations-Dateien
  - `lib`
    - Zusätzliche Bibliotheken
  - `lock`
    - Lock-File
  - `log`
    - Rollierende Log-Dateien
  - `tmp`
    - Temporäre Dateien, die bei einem gestoppten Server gelöscht werden können

# Dockerfiles zur Erstellung einer einfachen Artemis-Umgebung

 **apache** / **activemq-artemis**


Used by ▾ 6


Watch ▾ 52


★ Star 504


Fork 507


<> Code

 Pull requests 24

 Actions

 Projects 0

 Security

 Insights


Branch: master ▾ **activemq-artemis** / **artemis-docker** /

Create new file






Upload files


Find file

History

 **franz1981** and **clebertsuconic** ARTEMIS-2245 Cleaning up Docker images preparation, run and docs Latest commit c77e64a on 8 Feb

..

 <a href="#">Dockerfile-centos</a>	ARTEMIS-2245 Cleaning up Docker images preparation, run and docs	8 months ago
 <a href="#">Dockerfile-ubuntu</a>	ARTEMIS-2245 Cleaning up Docker images preparation, run and docs	8 months ago
 <a href="#">docker-run.sh</a>	ARTEMIS-2245 Cleaning up Docker images preparation, run and docs	8 months ago
 <a href="#">prepare-docker.sh</a>	ARTEMIS-2245 Cleaning up Docker images preparation, run and docs	8 months ago
 <a href="#">readme.md</a>	ARTEMIS-2245 Cleaning up Docker images preparation, run and docs	8 months ago

 **readme.md**

## Docker Image Example

This is an example on how you could create your own Docker Image For Apache ActiveMQ Artemis based on CentOS or Ubuntu.



1.4

## **UPGRADE EINER ARTEMIS-INSTALLATION**

- Durch die Trennung der Server-Binaries und der Broker-Konfiguration ist ein Upgrade des Servers relativ einfach
- Die Artemis-Community verspricht die Abwärts-Kompatibilität neuer Server-Versionen zu existierenden Konfigurationen
  - Alternativ werden in den Versionshinweisen detaillierte Anweisungen zur Migration gegeben
- Die verwendeten Server-Binaries sind in der `artemis.profile` hinterlegt
  - Ein Server-Upgrade ist damit nichts anderes als eine Anpassung der `ARTEMIS_HOME`-Property auf die neue Distribution

- Das Konfigurationsverzeichnis ist in einem Versionsverwaltungssystem abzulegen
  - Das bietet unabhängig vom Upgrade eigentlich nur Vorteile
    - Revisionssicherheit
    - Staging
    - Zentraler Zugriff auf die Konfigurationsdateien zum Ausrollen eines Brokers

- Die Artemis-Konfiguration wird über ein Docker-Image ausgeliefert
  - Die Binaries werden über ein Host-Mounting eingebunden
- Die Artemis-Binaries sind zusammen mit einem Broker-Konfigurationsverzeichnis in einem Docker-Image verpackt
  - `etc` wird als Volume oder Host-Mounting eingebunden
- Ein Binary-Docker-Image wird mit einem Broker-Konfigurationsimage verlinkt
  - z.B. mit `docker-compose`

2

## KONFIGURATION

2.1

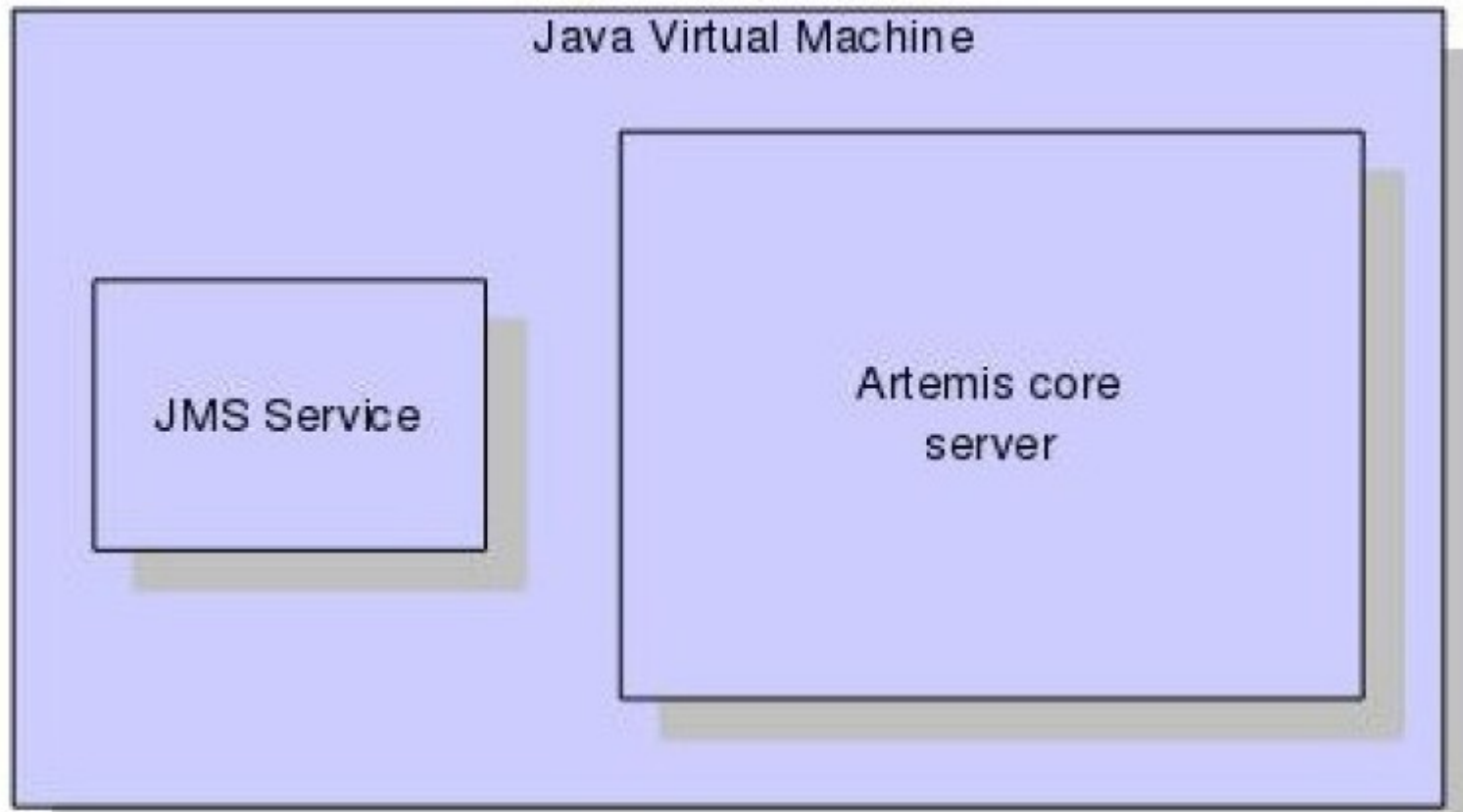
## ÜBERSICHT

- Die durch das artemis-Skript erzeugte Broker-Konfiguration enthält im Verzeichnis `etc` die Konfigurationsdateien
  - `artemis.profile`
  - `artemis-roles.properties`
  - `artemis-users.properties`
  - `bootstrap.xml`
  - `broker.xml`
  - `jolokia-access.xml`
  - `login.config`
  - `logging.properties`
  - `management.xml`

2.2

**JAVA**





- `artemis.profile`
- Speicherkonfiguration der Java Virtual Machine
  - Heap
  - Stack
  - Garbage Collection
- Logging in der `logging.properties`
  - Log-Dateien werden in das `log`-Verzeichnis geschrieben
    - `artemis.log`
    - `audit.log`

- `login.config`
  - Eine Standard JAAS-Konfiguration
- Standard-Konfiguration benutzt die beiden Dateien
  - `artemis-users.properties`
  - `artemis-roles.properties`
- Auch andere Konfigurationen sind möglich
  - Insbesondere LDAP
  - Beispiele unter  
`https://github.com/apache/activemq-artemis/blob/master/tests/integration-tests/src/test/resources/login.config`

```
LDAPLogin {  
    org.apache.activemq.artemis.spi.core.security.jaas.LDAPLoginModule  
    required  
        debug=true  
        initialContextFactory=com.sun.jndi.ldap.LdapCtxFactory  
        connectionURL="ldap://localhost:1024"  
        connectionUsername="uid=admin,ou=system"  
        connectionPassword=secret  
        connectionProtocol=s  
        authentication=simple  
        userBase="ou=system"  
        userSearchMatching="(uid={0}) "  
        userSearchSubtree=false  
        roleBase="ou=system"  
        roleName=cn  
        roleSearchMatching="(member=uid={1},ou=system) "  
        roleSearchSubtree=false  
    ;  
};
```

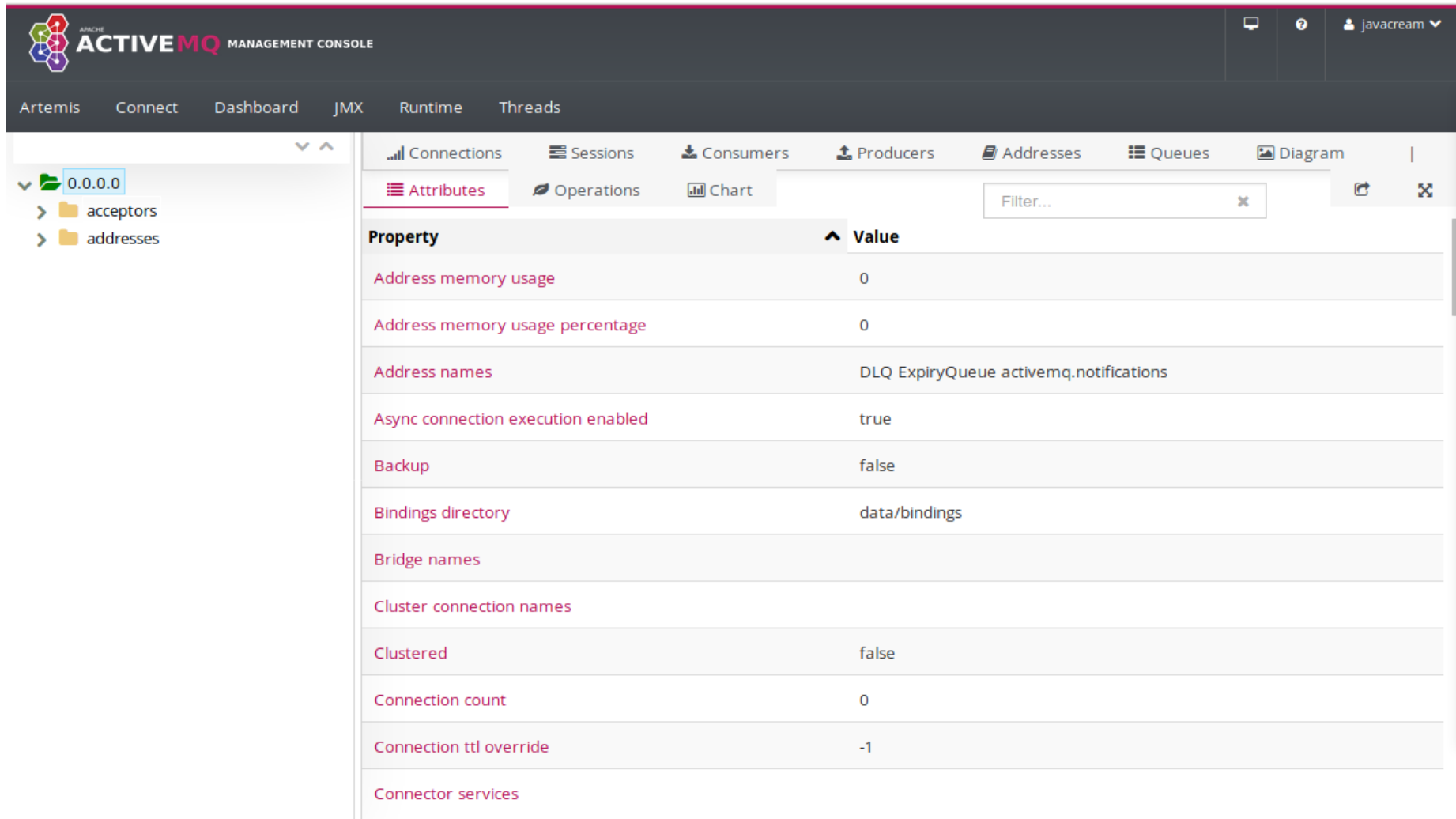
- `bootstrap.xml`
- Die eigentliche Broker-Konfiguration
  - `broker.xml`
- Artemis Web-Anwendungen
  - Zugriff über <http://localhost:8161>
  - Basiert auf Hawt (<http://hawt.io>)
    - Web-Konsole über `/console`
    - `artemis-plugin`
      - Hawt-Zugriff auf Artemis-JMX-Objekte
    - `artemis-branding`
      - Artemis Branding der Hawt-Anwendung

2.3

## **BASIS-ÜBERWACHUNG**

- JMX
- Jolokia-basiert
  - "Jolokia is remote JMX with JSON over HTTP"
  - <https://jolokia.org/>
- Security-Konfiguration unter jolokia-access.xml

# Die Artemis-Webkonsole



The screenshot shows the Apache Artemis Management Console interface. The top navigation bar includes 'Artemis', 'Connect', 'Dashboard', 'JMX', 'Runtime', and 'Threads'. The left sidebar shows a tree view with '0.0.0.0' expanded, containing 'acceptors' and 'addresses'. The main content area displays the 'Attributes' tab for a selected queue. The table lists various properties and their values.

Property	Value
Address memory usage	0
Address memory usage percentage	0
Address names	DLQ ExpiryQueue activemq.notifications
Async connection execution enabled	true
Backup	false
Bindings directory	data/bindings
Bridge names	
Cluster connection names	
Clustered	false
Connection count	0
Connection ttl override	-1
Connector services	



2.4

## **BERECHTIGUNGSKONZEPT ZUM ZUGRIFF AUF RESSOURCEN**

- `management.xml`
- Kontrolle des Zugriffs für
  - Web Console
  - Management Interface

2.5

## KONFIGURATION IM DETAIL

Type to search

- [Introduction](#)
- [Legal Notice](#)
- [Preface](#)
- [Project Info](#)
- [Versions](#)
- [Messaging Concepts](#)
- [Architecture](#)
- [Using the Server](#)
- [Upgrading](#)
- [Address Model](#)
- [Protocols and Interoperability](#)
- [AMQP](#)
- [MQTT](#)
- [STOMP](#)
- [OpenWire](#)
- [Core](#)
- [Mapping JMS Concepts to the Core ...](#)



A



## Apache ActiveMQ Artemis User Manual

The User manual is an in depth manual on all aspects of Apache ActiveMQ Artemis



3

# ÜBERSICHT DES PROGRAMMIERMODELLS

3.1

## **MESSAGING**

Messaging

- Artemis selbst ist keine direkte Implementierung eines JMS-Servers
  - Das ist bereits an der Administration von Destinationen wie Queues und Topics ersichtlich
- Die JMS-Unterstützung wird Client-seitig realisiert
  - Ein JMS-Wrapper setzt die JMS-Befehle in das native Artemis-API um



- Über "Acceptors" kann der Artemis-Server neben dem nativen Protokoll durch weitere Protokolle angesprochen werden:
  - AMQP
    - Advanced Message Queuing Protocol
    - Spezifikation der OASIS
  - MQTT
    - Ein leichtgewichtiges Publish-Subscribe-Modell
    - Ebenfalls spezifiziert durch OASIS
  - STOMP
    - Simple Text Oriented Messaging Protocol
  - OpenWire
    - Das native ActiveMQ-Protokoll
      - Damit können ActiveMQ-Clients (Version  $\geq 5$ ) direkt mit dem Artemis-Server kommunizieren

3.2

## **TEST-ANWENDUNGEN**

## Apache ActiveMQ Artemis Examples

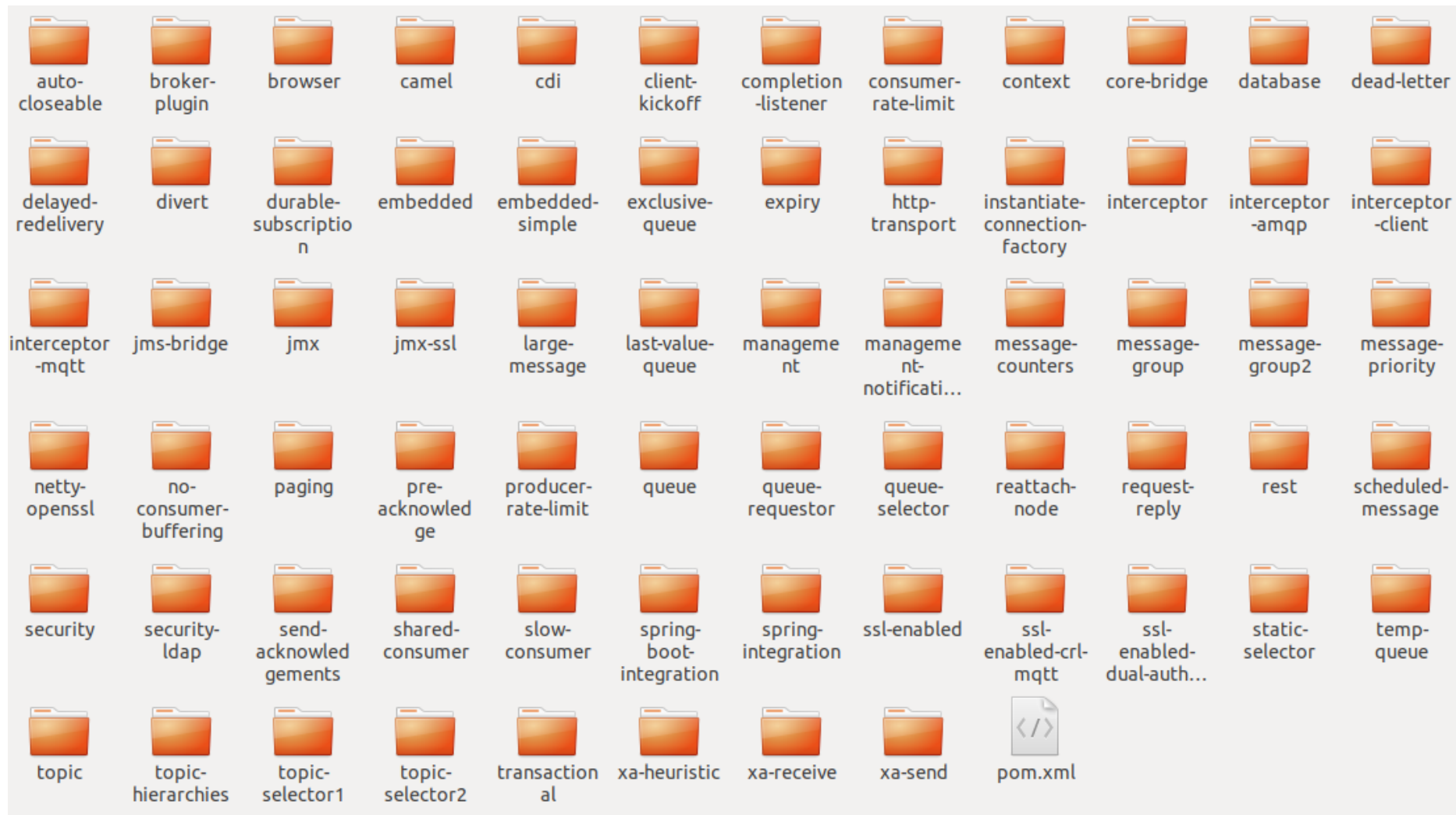
Apache ActiveMQ Artemis comes with over 90 runnable examples. These can be found in the examples directory in the root of the distribution folder. There are examples covering JMS as well as all the protocols and functionality that Apache ActiveMQ Artemis supports.

Each example has its own instructions as to how they can be run, but for most of them it is as simple as running `mvn verify` from the example directory. This will start a broker with the correct configuration, run the example and then stop the broker. You'll need to ensure there is not a broker already running as this may conflict with the broker that is configured and used in the example.

You may also run the example without starting the server by specifying `-PnoServer` (`mvn -PnoServer verify`). You can then use your own server for the example.

Notice that some examples will require special cases and multiple servers, for example in some Failover tests. Always refer to the README on each example.

- Die Beispielanwendungen sind größtenteils mit Java realisiert
- Zum Erzeugen der ausführbaren Test-Anwendung sowie zum Starten des jeweiligen Tests wird Apache Maven als Build-Werkzeug eingesetzt
  - `pom.xml` im jeweiligen Example-Unterverzeichnis
  - `mvn verify`
    - Starten mit embedded Artemis Server
    - Vorsicht: Ein externer Artemis-Server mit Standard-Ports darf nicht parallel dazu gestartet sein
  - `mvn -PnoServer verify`
    - Benutzung eines externen Artemis Servers
- Die für die Beispiele jeweils notwendige Broker-Konfiguration wird durch ein Skript erzeugt, das im `target`-Verzeichnis abgelegt wird
  - Hier ist im Standard über die Option `--noWeb` die Web Konsole aber auch Jolokia deaktiviert
    - Der Broker kann aber nach Anpassung des Skripts jederzeit neu erzeugt werden



## 4

# MONITORING IM DETAIL

4.1

**JMX**

- Sämtliche Ressourcen von Artemis sind über JMX abgreifbar
  - Broker
  - Queues
  - Producer und Consumer
  - Connections
- Zugriff über
  - Web Konsole
    - Hübsche und intuitiver Zugriff auf relevante Informationen
  - Jolokia REST-API
    - Technische Sicht
      - ObjectName
      - Attributes
      - Operations



- Nach Authentifizierung können Zugriffe auf das JMX-System auch direkt über eine RESTful-Syntax erfolgen
- Base-URL ist <http://localhost:8161/console/jolokia>
  - Leseoperationen mit `/read/<JMX-ObjectName>/<Attribute>`
  - `/read/java.lang:type=Memory/HeapMemoryUsage/used`
    - ```
{"request":  
  {"path":"used","mbean":"java.lang:type=Memory","attribute":"HeapMemoryUsage","type":"read"},"value":278251520,"timestamp":1572546842,"status":200}
```
- Damit ist die Integration von Jolokia in andere Überwachungssysteme sehr einfach

4.2

## **TROUBLE SHOOTING**

- Mehrere Ebenen
  - Java Virtual Machine
  - Networking
  - Messaging
  - Message Persistence

- Hauptsächlicher Grund für einen potenziellen Fehler ist die Speicherverwaltung und die Garbage Collection
  - Stress-Situationen durch hohe Last
  - Memory Leaks
- Überwachung durch
  - `java.lang:type=Memory`
  - Zur Nachvollziehbarkeit auch stets das GC-Logfile

- Artemis stellt die Durchsätze auf den einzelnen Acceptors, Connections und Destinations bereit
- Zusätzlich muss allerdings die normale Netzwerk-Überwachung im Rahmen eines Application Performance Managements eingesetzt werden
  - Artemis ist einfach als Messaging-System in Standard-APM-Werkzeuge wie AppDynamics zu integrieren

- Die durch die Message-Destinations definierten Kommunikationskanäle eignen sich hervorragend zur Erstellung aussagekräftiger Metriken
  - Anzahl der Messages
  - Größe und Durchsatz
  - Anzahl Antworten aufgeteilt auf Ergebnisse und Fehler
- Das vom Messaging Broker unterstützte Programmiermodell ist sehr gut geeignet, auch fachliche Fehlersituationen sinnvoll zu gruppieren:
  - Transaktionsabbrüche für temporäre Fehlersituationen, die potenziell neu aufgesetzt werden können
  - Acknowledgement-Strategien
- Infrastruktur
  - Expiration
  - Dead Letter Queue
  - Skalierung
  - Failover

- Artemis schreibt eine Vielzahl von Informationen permanent in das Dateisystem
- Damit ist ein schnelles Dateisystem verpflichtend
  - Ein langsames Dateisystem ist nach aller Erfahrung eines der ersten Bottlenecks, die innerhalb einer Performance-Analyse von Artemis erkannt wird
- Eine Überwachung der Größe der Journal-Dateien ein einfaches Mittel, Probleme zu erkennen
  - In den Zeiten von JMX und Hawt etwas antiquiert, aber immer noch im Einsatz

- Ablage der Messages sowie der Listen von Konsumenten und Produzenten
- Zwei Strategien
  - Dateibasiertes Journal
  - JDBC-Datenbank, von Artemis getestete und unterstützte System sind
    - PostgreSQL 9.4.x
    - MySQL 5.7.x
    - Apache Derby 10.11.1.1



- Journal
  - Standard-Mechanismus bei Artemis
  - Hoch-optimierte Implementierung des Schreibens der Messages
  - Die Anforderungen an das zugrunde liegende Dateisystem sind hoch
- JDBC
  - Für einige Unternehmen die geeignete Wahl, um insbesondere auch Clusterlösungen aufzusetzen
    - Auf Datenbanken basierende Lösungen sind in System-Architekturen sehr etabliert
  - Allerdings in der Summe deutlich langsamer als das Journal
- Beide Varianten haben deshalb ihre Vorteile
- Konfiguration und Optimierung
  - [https://activemq.apache.org](https://activemq.apache.org/components/artemis/documentation/latest/persistence.html) /components/artemis/documentation/latest/persistence.html

5

# SYSTEM-ARCHITEKTUREN

5.1

## **AUSFALLSICHERHEIT UND LASTVERTEILUNG**

- Im Wesentlichen ein Aktiv-Passiv-System
  - <https://activemq.apache.org/components/artemis/documentation/latest/ha.html>
- Strategien
  - Shared Store
  - Replikation
  - None (kein Backup)

- Ein Server sendet beim Herunterfahren seine Daten kontrolliert an einen anderen Server

5.2

## **CLUSTER**

- Partitionierung und Bridge-Lösungen
  - <https://activemq.apache.org/components/artemis/documentation/latest/core-bridges.html>
- Federation
  - <https://activemq.apache.org/components/artemis/documentation/latest/federation.html>
- Lastverteilung
  - <https://activemq.apache.org/components/artemis/documentation/latest/clusters.html>

5.3

## **HARDENING**



- Die Schritte einer Standard-Installation auf einem Linux-System ist beispielsweise durch die Dockerfiles beschrieben
- Alle nicht benötigten Protokolle sind zu deinstallieren
- JMX-Protokolle und Web Konsole auf jeden Fall gesondert absichern
  - am Besten über die Bindung an ein internes Admin-Netzwerk

WHITE PAPER



## A Pentesters Guide to Hacking ActiveMQ-Based JMS Applications