Message Broker

For example I have an application1 (python or java or Dot net application) which wants to communicate with another application2(python or java or Dot net application).

Message broker is a software that enables applications or systems or services or micro services to communicate with each other and exchange information.

* Why do applications need to exchange information?

For an E-commerce application there will be multiple projects as

Shopping cart

Tracking the product

Product reviews

Payment gateway ,… etc…

If we build the application in single module, then we face difficulty in deploying the application in any bug scenarios. i.e,. we build apps as microservices.

In between applications to exchange the information as notifications we use message broker.

The Open-source message brokers are

1. Active mq supports queue and topic based.
2. Rabbit mq supports queue based.
3. Apache Kafka supports topic based.

Queue and Topic

A queue means a message goes to one and only one possible subscriber.

A topic goes to each and every subscriber. Topics are for the publisher-subscriber model, while queues are for point-to-point. A JMS topic is the type of destination in a 1-to-many model of distribution.

Message1 : if I want to send the message to one application I will use queue.

If I want to send the message to multiple applications I will topic.

**ActiveMQ**

Application1 can send a message from python application.

Application2 can receive the message from java application.

Link: <https://www.tecmint.com/install-apache-activemq-on-centos-rhel/>

Setting up of ActiveMq in RHEL8 linux

1. Create an instance in your cloud environment with RHEL8.
2. Login using created pem file.
3. Install java
4. # dnf update
5. # dnf install java-11-openjdk-devel

Java -version

Text

Description automatically generated

Once Java installed, you can proceed further to download the latest version of ActiveMQ or use the following wget command to grab the source package

# cd /opt

# wget <http://archive.apache.org/dist/activemq/5.16.2/apache-activemq-5.16.2-bin.tar.gz>

# tar -xvzf apache-activemq-5.16.2-bin.tar.gz

# cd apache-activemq-5.16.2

To run **ActiveMQ** as a service, you need to create an **ActiveMQ** service unit file under the user called **ActiveMQ**, so start by creating the user using useradd command

# useradd activemq

# chown -R activemq:activemq /opt/apache-activemq-5.16.2

# ls -l /opt/apache-activemq-5.16.2/

# vi /etc/systemd/system/activemq.service

Add the following configuration in the ActiveMQ. Service file.

[Unit]

Description=Apache ActiveMQ Message Broker

After=network-online.target

[Service]

Type=forking

User=activemq

Group=activemq

WorkingDirectory=/opt/apache-activemq-5.16.2/bin

ExecStart=/opt/apache-activemq-5.16.2/bin/activemq start

ExecStop=/opt/apache-activemq-5.16.2/bin/activemq stop

Restart=on-abort

[Install]

WantedBy=multi-user.target

# systemctl daemon-reload

# systemctl start activemq.service

# systemctl enable activemq.service

# systemctl status activemq.service

If the service is not working open the ports required in 8161 in created ec2 machine

Edit the jetty.xml file with accessing all IP’s or required IP’s.

<bean id="jettyPort" class="org.apache.activemq.web.WebConsolePort" init-method="start">

<!-- the default port number for the web console -->

<property name="host" value="0.0.0.0"/>

<property name="port" value="8161"/>

</bean>

And restart the service.

Access with

<http://IP> address:8162/admin