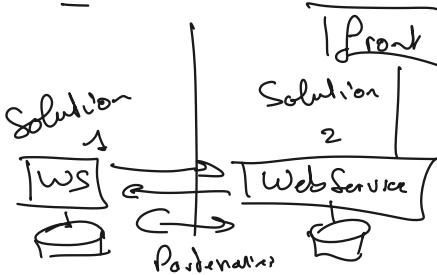
Bonjour tout le Monde

Architectures Orientés

Messages

- Middle ware
  - Oriented
  - Message

3 Tiers



- obsolète
  → API Rest

WS → Appels synchrones

Appel Synchrone →

- Temps réel → (+)
- Que se passe t'il si la Transaction est interrompue -
  - ↳ besoin de relancer ....
    - → pas dispo dans cette architecture -

→ Une opération doit pouvoir être faite intégralement ou pas du tout

Définition d'une Transaction cohérente -

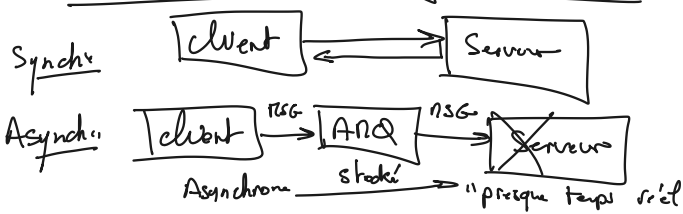
↓

Equivalent → "Coup de fil"

Alternative → (- WhatsApp
- SMS
- -)

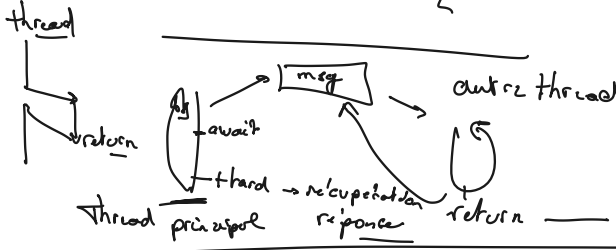
A NQ

Kafka —

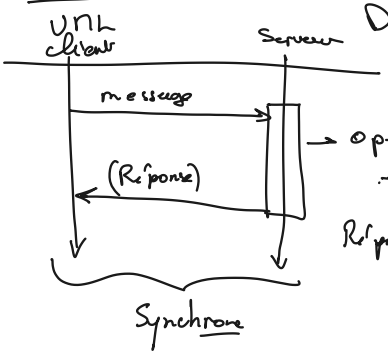


Prog Asynchrone en java, Python, .net, javascript, etc....

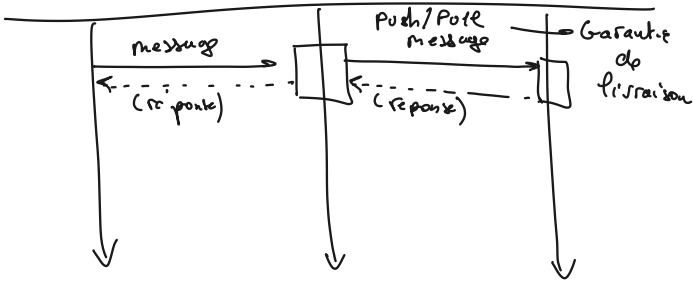
await fct(-) → async int fct(-)
{



Concept des Messages :



→ opération
- message : 0..n
clef / valeur
Réponse = 0..1 valeur
(simple / complexe)



Active MQ - permet la réponse aux messages (message de réponse corrélé)

client A → plutôt émetteur → pourrait aussi recevoir

client B → plutôt récepteur → pourrait aussi émettre

Métaphore → Coup de fil / SMS (peer to peer)

→ BroadCast

→ 1 Emetteur - n Récepteurs.

Publisher (PubSub) Subscribers

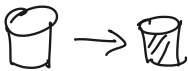
↳ pas de réponse.

1 msg = reçu n fois par les n clients

Relationnel vs NoSQL

Oracle
NSQL
Postgres
MariaDB
MySQL

→ HA → master / slave -



Up Down

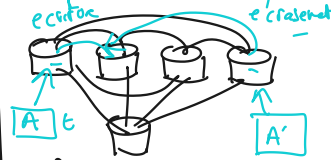
Réplication

→ Réplication asynchrone
→ risque de perte sur Bascule -

→ équiv. ActiveMQ / Artemis
→ forte cohérence
→ cluster limité (HA "difficile")

→ MongoDB
Cassandra
Redis

→ objets → fabrication façon cluster



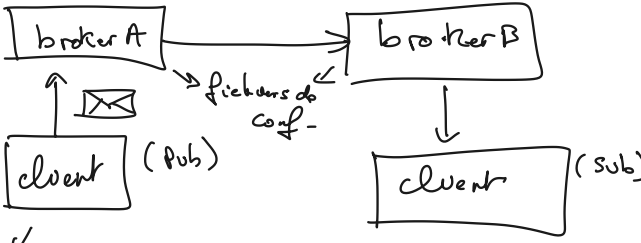
→ facile, pas cher

Plus: asynchronisme sur toute ligne

→ Équiv Kafka
→ forte HA
→ Risques par rapport à la cohérence (écrasement, perte, ...)

ActiveMQ en synchrone est une interface RPC / (LPC → local → method locale)

Remote Procedure Call

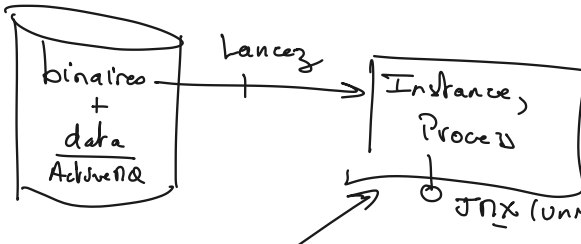


- ligne de cmd
- Java (Console)

Java Management eXtension

↓

Gestion



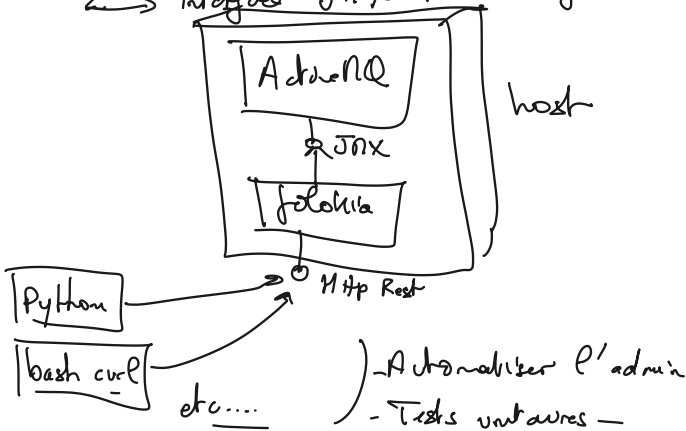
- Piloter (arrêter, vider une file, ...)
- Consulter ... métriques
- etc...

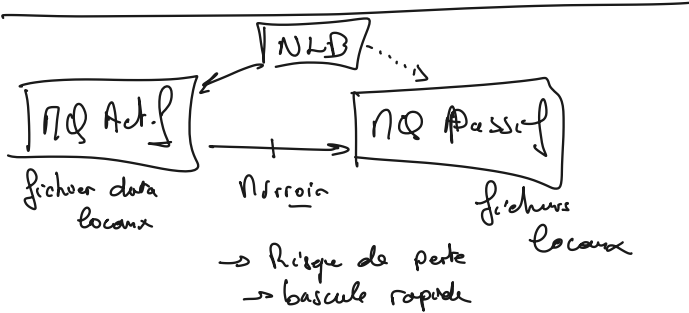
- jConsole (standard dans un JDK)

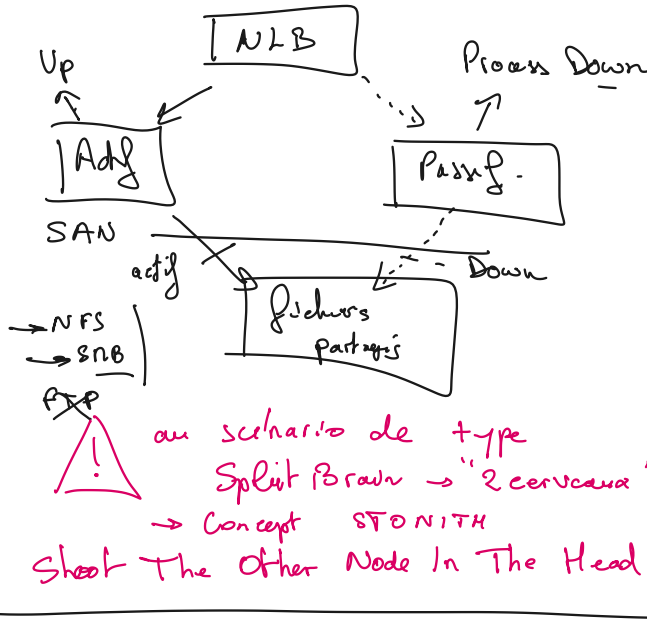
Pilotage de ActiveMQ :

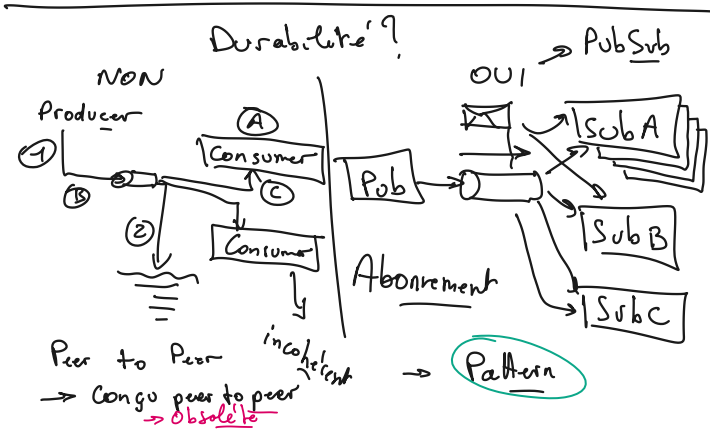
- Ligne de Commande locale (nécessite SSH)

- JMX
  → interface JMX → Rest via Jolokia



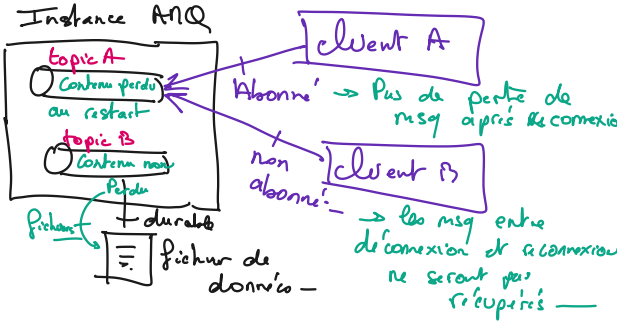



NLB
Up
Process Down
Actif
Passif
SAN
actif
Down
Disques partagés
NFS
SMB
au scénario de type
Split Brain → "2 cervaux"
→ Concept STONITH
Shoot The Other Node In The Head


Durabilité ?
→ PubSub
NON
OUI
Producer
A
Consumer
B
C
Consumer
Pub
Abonnement
SubA
SubB
SubC
Peer to Peer
→ Congo peer to peer
→ Obsolète
incohérent
→ Pattern

1. Point-To-Point / **Queue**
   a. Master publie un message dans une queue
   b. Un Receiver reçoit un message
   c. Il doit l'acquitter pour le supprimer de la queue, pour ne pas le recevoir à nouveau
2. Publish-Subscribe / **Topic**
   a. Publisher envoie un message
   b. Le message est supprimé quand tous les abonnés l'ont reçu.

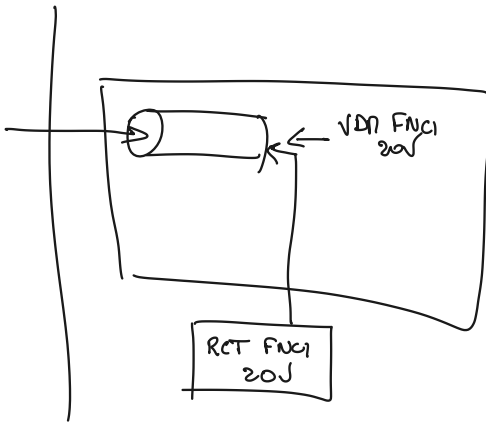
Il y a l'abonnement à la connexion, et l'abonnement durable (persistant)

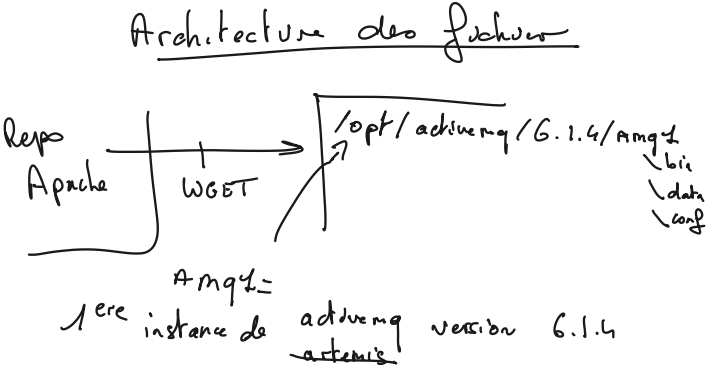


wget https://www.apache.org/dyn/closer.cgi?filename=/activemq/6.1.2/apache-activemq-6.1.2-bin.tar.gz&action=download

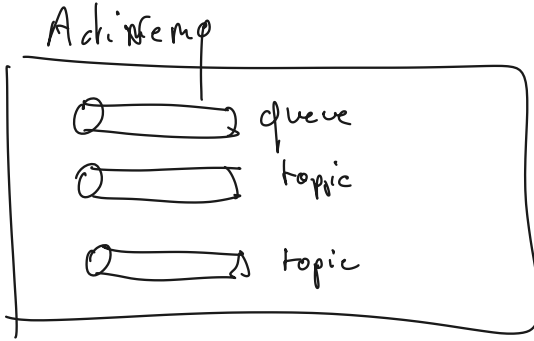
mv closer.cgi\?filename\=%2Factivemq%2F6.1.2%2Fapache-activemq-6.1.2-bin.tar.gz apache-activemq-6.1.2-bin.tar.gz

cd /opt
tar xf <dossier>/apache.....tar.gz


VULN FUNC
RET FUNC



App Web par défaut :
http://127.0.0.1:8161/index.html
admin / admin



Exercice :
Téléchargez, décompressez, lancez (activemq console) un serveur mq, et consulter son portail admin, envoyez un message de test via l'interface.

- sudo mkdir /opt/amq
- sudo chown student:student /opt/amq
- wget https://dlcdn.apache.org//activemq/6.1.2/apache-activemq-6.1.2-bin.tar.gz

Solution :

```
[student@trainer opt]$ sudo mkdir /opt/amq
[student@trainer opt]$ sudo chown student:student /opt/amq
[student@trainer opt]$ ll amq
total 0
[student@trainer opt]$ ll
total 4
drwxr-xr-x.  7 root    root     104 20 avril  2021 amd
drwxr-xr-x.  2 student student    6 10 juin  13:09 amq
...
[student@trainer opt]$ cd amq
[student@trainer amq]$ wget https://dlcdn.apache.org//activemq/6.1.2/apache-activemq-6.1.2-bin.tar.gz
--2024-06-10 13:10:31--  https://dlcdn.apache.org//activemq/6.1.2/apache-activemq-6.1.2-bin.tar.gz
Résolution de dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connexion à dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : 54773055 (52M) [application/x-gzip]
Sauvegarde en : « apache-activemq-6.1.2-bin.tar.gz »

apache-activemq-6.1.2-bin.tar.gz            100%
[=========================================================================
====================>]  52,24M   171MB/s    ds 0,3s

2024-06-10 13:10:32 (171 MB/s) — « apache-activemq-6.1.2-bin.tar.gz » sauvegardé
[54773055/54773055]

[student@trainer amq]$ ll
total 53492
-rw-rw-r--. 1 student student 54773055 11 avril 19:42 apache-activemq-6.1.2-bin.tar.gz
[student@trainer amq]$ tar xf apache-activemq-6.1.2-bin.tar.gz
[student@trainer amq]$ ll
total 53492
drwxr-xr-x. 10 student student      192 11 avril 17:35 apache-activemq-6.1.2
-rw-rw-r--.  1 student student 54773055 11 avril 19:42 apache-activemq-6.1.2-bin.tar.gz
[student@trainer amq]$ mkdir -p 6.1.2/amq1
[student@trainer amq]$ cd 6.1.2/amq1/
[student@trainer amq1]$ mv /opt/amq/apache-activemq-6.1.2/* .
[student@trainer amq1]$ ll
total 11148
-rwxr-xr-x. 1 student student 11354996 11 avril 17:35 activemq-all-6.1.2.jar
drwxrwxr-x. 4 student student      130 10 juin  13:10 bin
drwxr-xr-x. 2 student student     4096 10 juin  13:10 conf
drwxr-xr-x. 2 student student       26 10 juin  13:10 data
```

sudo yum install java-17-openjdk-devel
sudo alternatives --config java
[student@trainer bin]$ sudo alternatives --config java

Il existe 3 programmes qui fournissent « java ».

```
  Sélection    Commande
-----------------------------------------------
*  1           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.252.b09-2.el8_1.x86_64/jre/bin/java)
 + 2           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-
openjdk-11.0.13.0.8-4.el8_5.x86_64/bin/java)
   3           java-17-openjdk.x86_64 (/usr/lib/jvm/java-17-
openjdk-17.0.1.0.12-2.el8_5.x86_64/bin/java)
```

*Entrez pour garder la sélection courante [+] ou saisissez le numéro de type de sélection :**3***
[student@trainer bin]$ ./activemq console
-> Choisir java 17
./activemq console doit être OK

Retouche pour rendre l'admin accessible :
dans /conf/jetty.xml :

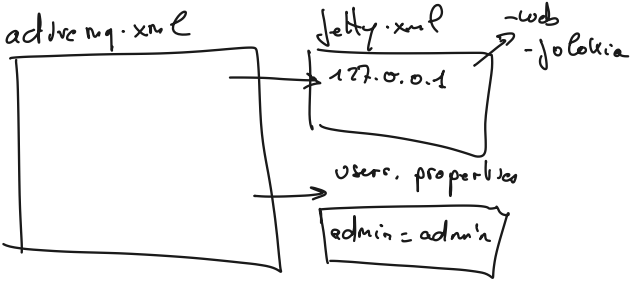
```
<bean id="jettyPort" class="org.apache.activemq.web.WebConsolePort" init-
method="start">
        <!-- the default port number for the web console -->
    <property name="host" value="0.0.0.0"/>
    <property name="port" value="8161"/>
  </bean>
```

ET

[student@trainer bin]$ cat ../conf/users.properties

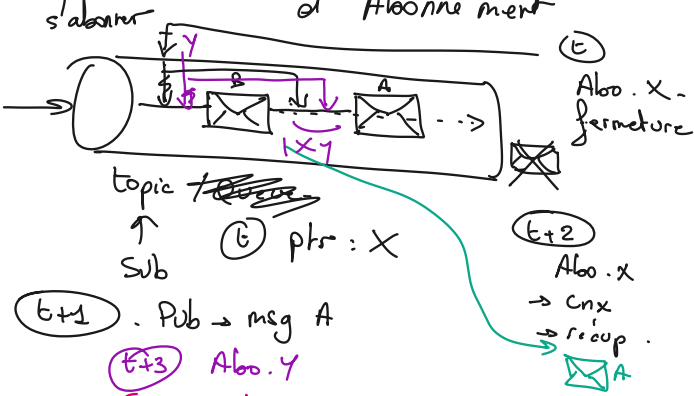
```
## ---------------------------------------------------------------------
## Licensed to the Apache Software Foundation (ASF) under one or more
## contributor license agreements.  See the NOTICE file distributed with
## this work for additional information regarding copyright ownership.


admin=Pa$$w0rd2424
```

Architecture par défaut
- non executable via activemq start



---

fonctionnement du principe d'Abonnement



Destination = chaîne

ex: topic1

alertes

. app1/ layer1/ alerts

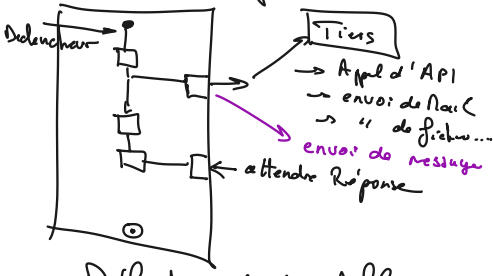
app1 . layer1. alerts

messages de app1.*

Notion de Corrélation

Exemple

Moteur de Workflow

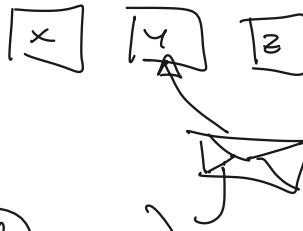


→ Définition de workflows

Instances de workflows

Exemple processus de passage d'une commande client

Demandes



Réponse

id Corr = (Y)

Texte = ~~~~~

Table clients

clients

idclient	nom	
1	A	
2	B	

mails
~~messages~~

idmail	idclient	msg
1	1	
2	1	
3	2	

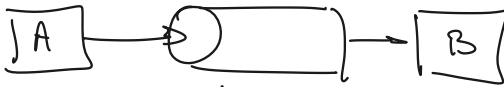
GUID

msg id = ID

msg Corr id = (2 | 1)

ID mail ID cli.

Cas (1) : Comm Type "Fire and forget"

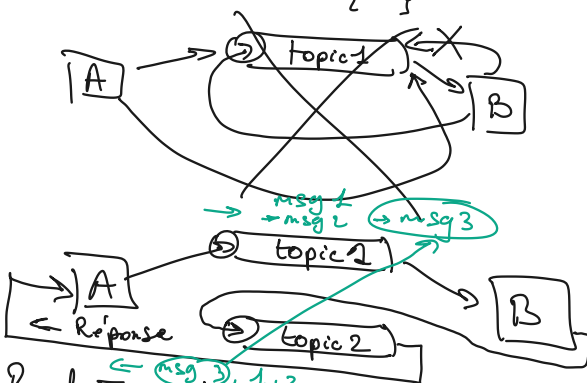


msg → body
(metadonnés)

equivaut à void operation(params)

Cas (2) : Comm type "Requête - Reponse"

Equivaut à Message operation (Message)
{ }



ReplyTo = topic2

Exemple
- msg = 1+2
- msg = 5+7
- Utiliser la Corrélation ID pour

Réponse :
12
3

Addition avec
- hash
- ID
- chaîne....
matcher

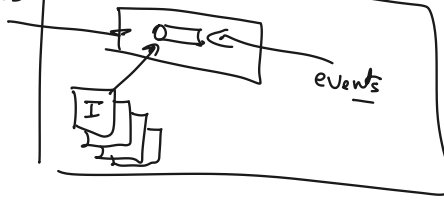
Pom.xml

→ broker integr

→ Pub

→ Sub

Hub



Site 1
(privatpool)

Site 2

client

SRW
VPN

broker

VPN

TCP

broker

Site 3

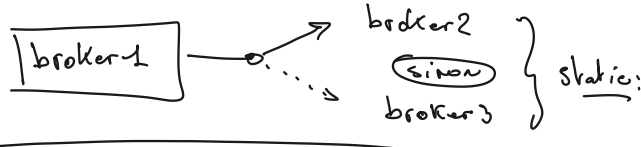
80
o
443

broker

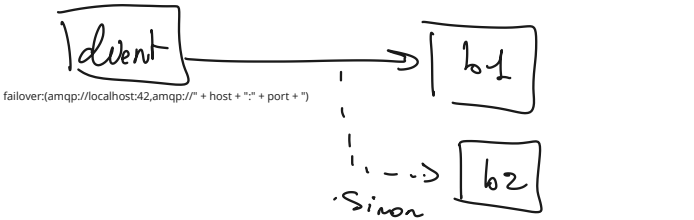
chat

https

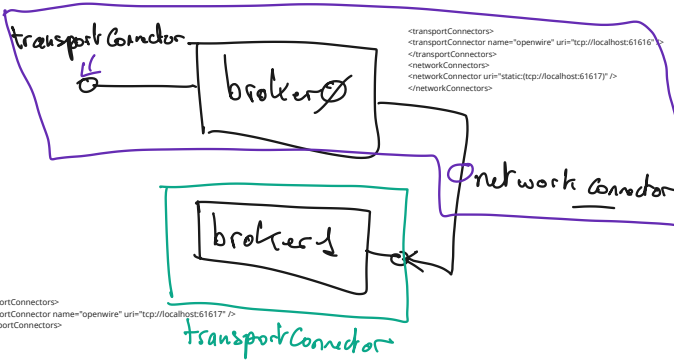
http(s)



multiples URIs du client (Pub / Sub)




```
failover:(amqp://localhost:42,amqp://" + host + ":" + port + ")
```




```
<transportConnectors>
<transportConnector name="openwire" uri="tcp://localhost:61616" />
</transportConnectors>
<networkConnectors>
<networkConnector uri="static:(tcp://localhost:61617)" />
</networkConnectors>
```

```
<transportConnectors>
<transportConnector name="openwire" uri="tcp://localhost:61617" />
</transportConnectors>
```

(1) Télécharger ANQ

↳ /opt/install/amq/6.1.4/bin

(2) Créer une instance :

/opt/install/amq/6.1.4/bin/activemq →

create /opt/amq/amq1

(3) → (re) toucher la conf

/opt/amq/amq1/conf/ ...

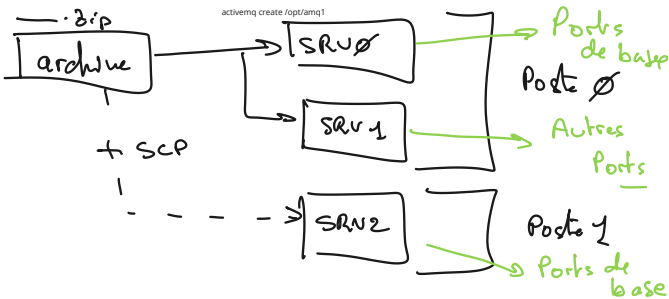
(4) démarrer / arrêter :

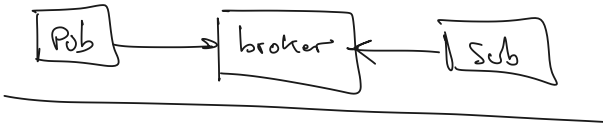
amq1 | start
stop

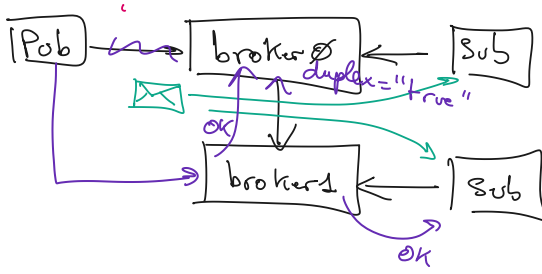
(5) Logs : tail -f activemq.log

Exercice : Fabriquez 3 instances

- 2 sur la 1ère machine



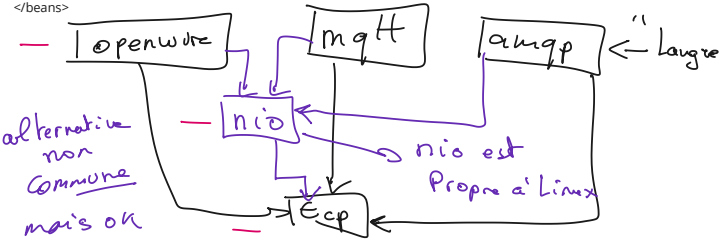




Exemple de conf simplifiée :

```
<beans xmlns="http://activemq.org/config/1.0">

 <broker brokerName="receiver" persistent="false" useJmx="false">
   <networkConnectors>
    <networkConnector uri="static:(tcp://localhost:62001)"/>
   </networkConnectors>

   <persistenceAdapter>
    <memoryPersistenceAdapter/>
   </persistenceAdapter>

   <transportConnectors>
    <transportConnector uri="tcp://localhost:62002"/>
   </transportConnectors>
 </broker>

</beans>
```

Couche ISO

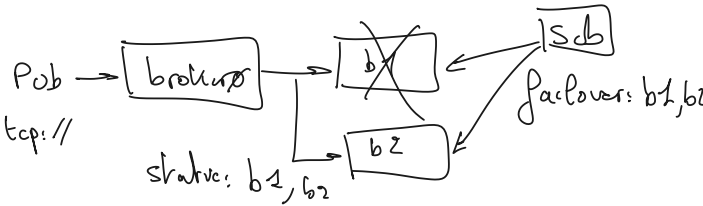


Modification des variables d'env plutôt que recompiler :

student@u24:/opt/install/amq-6.1.2/examples/amqp/java$ **export ACTIVEMQ_PORT=5772**
student@u24:/opt/install/amq-6.1.2/examples/amqp/java$ java -cp target/amqp-example-0.1-SNAPSHOT.jar example.Publisher
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Sent 1000 messages
Sent 2000 messages
Sent 3000 messages
Sent 4000 messages
Sent 5000 messages
Sent 6000 messages
Sent 7000 messages
Sent 8000 messages
Sent 9000 messages
Sent 10000 messages
student@u24:/opt/install/amq-6.1.2/examples/amqp/java$ **export ACTIVEMQ_PORT=5773**
student@u24:/opt/install/amq-6.1.2/examples/amqp/java$ java -cp target/amqp-example-0.1-SNAPSHOT.jar example.Publisher
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Exception in thread "main" jakarta.jms.JMSException: finishConnect(..) **failed: Connexion refusée: localhost/127.0.0.1:5773**
 at org.apache.qpid.jms.provider.ProviderException.toJMSException(ProviderException.java:34)
 at org.apache.qpid.jms.exceptions.JmsExceptionSupport.create(JmsExceptionSupport.java:80)
 at org.apache.qpid.jms.exceptions.JmsExceptionSupport.create(JmsExceptionSupport.java:112)

URI :
As the static transport protocol is for broker discovery, it should not be used by client programs. Clients wishing to failover to a static list of broker instances, should use the failover:// transport instead.

conf d'un broker vers une liste de brokers (un seul dans la liste :) : utiliser static:
conf d'un client vers une liste de brokers (un seul dans la liste :) : utiliser failover:



Administrer (requêter) activemq en ligne de commande :

```
./amq1
INFO: Using default configuration
     Configurations are loaded in the following order: /etc/default/activemq
/home/student/.activemqrc /opt/install/amq-6.1.2/bin/setenv

INFO: Using java '/usr/bin/java'
Extensions classpath:
 .....
Usage: /opt/install/amq-6.1.2/bin/activemq [--extdir <dir>] [task] [task-options] [task
data]

Tasks:
   browse               - Display selected messages in a specified destination.
   bstat                - Performs a predefined query that displays useful statistics
regarding the specified broker
   consumer             - Receives messages from the broker
   create               - Creates a runnable broker instance in the specified path.
   decrypt              - Decrypts given text
   dstat                - Performs a predefined query that displays useful tabular statistics
regarding the specified destination type
   encrypt              - Encrypts given text
   export               - Exports a stopped brokers data files to an archive file
   list                 - Lists all available brokers in the specified JMX context
   producer             - Sends messages to the broker
   purge                - Delete selected destination's messages that matches the message
selector
   query                - Display selected broker component's attributes and statistics.
   start                - Creates and starts a broker using a configuration file, or a broker
URI.
   stop                 - Stops a running broker specified by the broker name.
```

Obtenir de l'aide :

```
student@u24:/opt/amq/amq1/bin$ ./amq1 browse --help
INFO: Using default configuration
 Configurations are loaded in the following order: /etc/default/activemq
/home/student/.activemqrc /opt/install/amq-6.1.2/bin/setenv
...
Task Usage: Main browse [browse-options] <destinations>
Description: Display selected destination's messages.

Browse Options:
   --amqurl <url>              Set the broker URL to connect to. Default tcp://localhost:61616
   --msgsel <msgsel1,msglsel2>   Add to the search list messages matched by the query
similar to
                               the messages selector format.
   --factory <className>       Load className as the jakarta.jms.ConnectionFactory to use
for creating connections.
   --passwordFactory <className> Load className as the
org.apache.activemq.console.command.PasswordFactory
                               for retrieving the password from a keystore.
   --user <username>           Username to use for JMS connections.
   --password <password>       Password to use for JMS connections.
   -V<header|custom|body>      Predefined view that allows you to view the message
header, custom
                               message header, or the message body.
   --view <attr1>,<attr2>,...   Select the specific attribute of the message to view.
   --version                   Display the version information.
   -h,-?,--help                Display the browse broker help information.

Examples:
   Main browse --amqurl tcp://localhost:61616 FOO.BAR
      - Print the message header, custom message header, and message body of all
messages in the
        queue FOO.BAR

   Main browse --amqurl tcp://localhost:61
```

Exemples administratifs :

Parcourir le contenu d'une QUEUE (mais pas un TOPIC) :
student@u24:/opt/amq/amq1/bin$ ./amq1 browse –amqurl tcp://localhost:61616 -Vheader,body event
INFO: Using default configuration
    Configurations are loaded in the following order: /etc/default/activemq /home/student/.activemqrc /opt/install/amq-6.1.2/bin/setenv

I...
ACTIVEMQ_HOME: /opt/install/amq-6.1.2
ACTIVEMQ_BASE: /opt/amq/amq1
ACTIVEMQ_CONF: /opt/amq/amq1/conf
ACTIVEMQ_DATA: /opt/amq/amq1/data
JMS_BODY_FIELD:JMSText = Enter some text here for the message body...
JMS_HEADER_FIELD:JMSRedelivered = false
JMS_HEADER_FIELD:JMSType =
JMS_HEADER_FIELD:JMSDestination = event
JMS_HEADER_FIELD:JMSPriority = 0
JMS_HEADER_FIELD:JMSMessageID = ID:u24-37851-1718184014212-7:1:1:1:2
JMS_HEADER_FIELD:JMSTimestamp = 1718195245872
JMS_HEADER_FIELD:JMSExpiration = 0
JMS_HEADER_FIELD:JMSDeliveryMode = non-persistent
JMS_HEADER_FIELD:JMSCorrelationID =

FAQ > JMS > Can you browse a topic

You can browse queues, can you browse a topic?

No. But then consuming messages on a topic does not affect any other consumers, so you don't need to 'browse' per se, just subscribe.

Parcourir le contenu : il faut activer JMX
<broker useJmx="true" brokerName="BROKER1">

Configuration JMX :
https://activemq.apache.org/components/classic/documentation/jmx

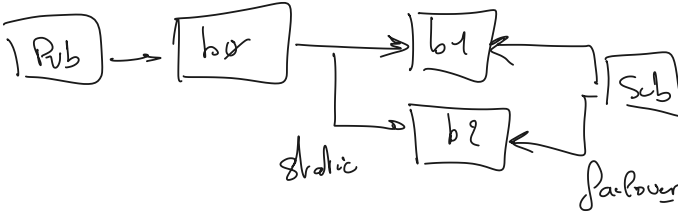
Exemple de requête administrative pour consulter via JMX en ligne de commande :
Pour cet exemple, le connecteur réseau n'est pas actif (il faudra configurer la sécurité) je passe donc par le PID pour faire simple :
(recherche de pid via ps -faux | grep amq1 - par exemple)

Recherche de tous les Topics :

```
student@u24:/opt/amq/amq1/bin$ ./amq1 query --pid 61035 -QTopic=* | grep Name
Name = ActiveMQ.Advisory.NetworkBridge
destinationName = ActiveMQ.Advisory.NetworkBridge
brokerName = amq1
Name = ActiveMQ.Advisory.Topic
destinationName = ActiveMQ.Advisory.Topic
brokerName = amq1
Name = event
destinationName = event
brokerName = amq1
Name = ActiveMQ.Advisory.Connection
destinationName = ActiveMQ.Advisory.Connection
brokerName = amq1
Name = ActiveMQ.Advisory.Queue
destinationName = ActiveMQ.Advisory.Queue
brokerName = amq1
Name = ActiveMQ.Advisory.MasterBroker
destinationName = ActiveMQ.Advisory.MasterBroker
brokerName = amq1
Name = ActiveMQ.Advisory.Producer.Topic.event
destinationName = ActiveMQ.Advisory.Producer.Topic.event
brokerName = amq1
```

Chaîne de connexion vers un activemq avec JMX remote actif :

```
–jmxurl service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
```



Persistance – msg enregistrés sur le disque ≠ pas perdu si restart

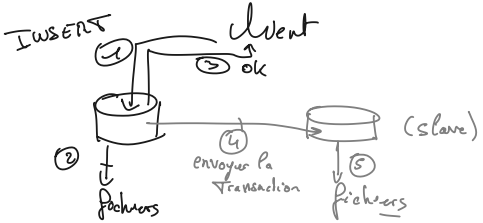
Durable (dans la doc)

Abonnement du client =

Persistance → Par file + Par message

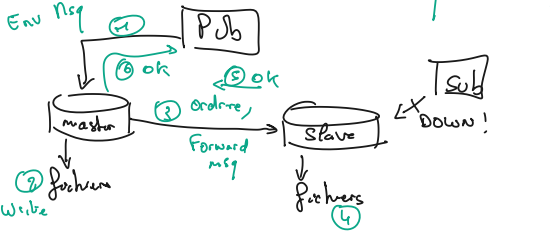
<kahaDB directory="activemq-data" journalMaxFileLength="32mb"/>

Réplication Asynchron (cas général en BDD)



↳ entre (3) et (4) → crash du master = Perte de Transactions

ActiveMQ (Synchrone)



Dusable ou Abonné (subscribed) → Un Abonné, ou Sub Durable, ne perdra pas ses messages après reconnexion

(pas juste Subscriber pas un Subscriber pas abonné)

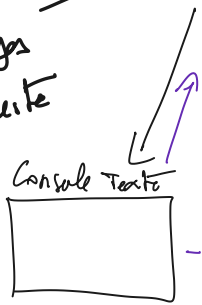
Persistance → pas de perte sur restart (par message)

↳ Dans la disque

1) Si pas de perte = msg persistant

topic = "alertes"

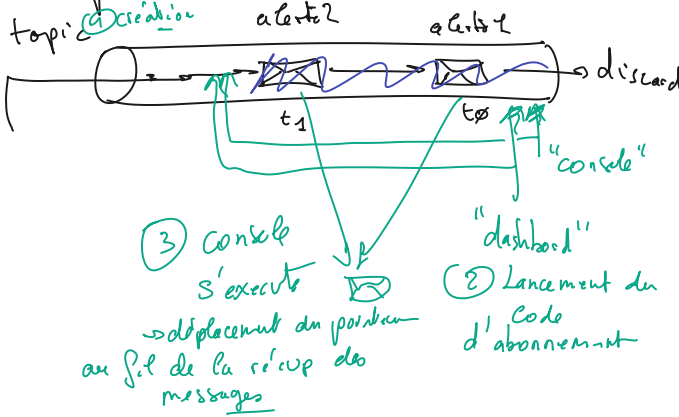
messages d'alerte

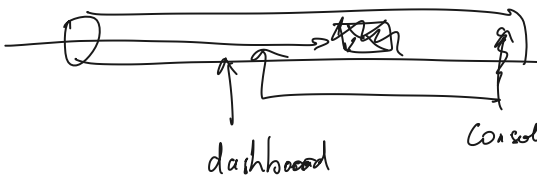


connection.setClientID("console");
Si pas déjà abonné :
MessageConsumer consumer = session.createDurableSubscriber(topic,"console");
-> quel identifiant pour quel topic

connection.setClientID("dashboard");
Si pas déjà abonné :
MessageConsumer consumer = session.createDurableSubscriber(topic,"dashboard");

Exemple dans le Temps :





Code (a tester) qui supprime un abonné (via JMX, forcément...)

```
import javax.management.MBeanServerConnection;
import javax.management.ObjectName;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.management.remote.JMXServiceURL;
import java.util.HashMap;
import java.util.Map;

public class DurableSubscriberManager {
   public static void main(String[] args) throws Exception {
      String brokerURL = "service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi";
      JMXServiceURL url = new JMXServiceURL(brokerURL);
      JMXConnector jmxc = JMXConnectorFactory.connect(url, null);
      MBeanServerConnection conn = jmxc.getMBeanServerConnection();

      ObjectName brokerName = new
ObjectName("org.apache.activemq:type=Broker,brokerName=amq1");
      String clientId = "myClientID";
      String subscriptionName = "dashboard";

      conn.invoke(brokerName, "destroyDurableSubscriber",
            new Object[]{clientId, subscriptionName},
            new String[]{String.class.getName(), String.class.getName()});
      jmxc.close();
   }
}
```