Execution de l'exemple :
Lancer le serveur :

- soit créer un dossier complet du serveur, et y copier la configuration
- Soit copier les fichiers .properties d'un serveur dans /opt/amq-in-action/src/main/resource/book/ch2 (a corriger)
- lancer le serveur avec activemq console -DactiveMQ.conf=<**dossier** des fichiers de conf>

Solution :
./activemq console -DactiveMQ.conf=/opt/amq-in-action-example-src/src/main/resources/org/apache/activemq/book/ch2

Lancer le producer :
java -cp <package>.jar (se trouve dans le dossier target) **<package de la classe>.<nom de la classe>**

mvn exec:java -Dexec.mainClass=org.apache.activemq.book.ch2.portfolio.Publisher
ou :
avec maven (apt install maven) en ligne de commande :
mvn exec:java -Dexec.mainClass=<package de la classe>.<nom de la classe> -Dexec.args="<paramètres>"

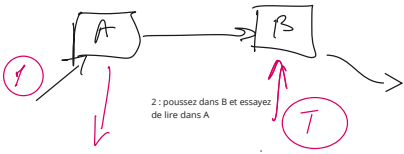
exemple : mvn exec:java \ -
Dexec.mainClass=org.apache.activemq.book.ch3.portfolio.Consumer \ -
Dexec.args="CSCO ORCL"

Lancer le consommateur :
mvn exec:java -Dexec.mainClass=org.apache.activemq.book.ch2.portfolio.Consumer

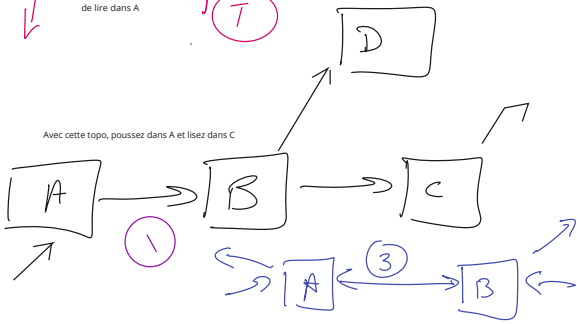
Voir un message être envoyé et reçu, consulter la console du serveur ActiveMQ

Linux, déplacement de tous les fichiers y compris cachés :
Exécuter préalablement :

shopt -s dotglob

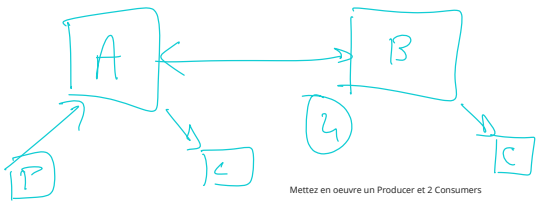


2 : poussez dans B et essayez de lire dans A



Avec cette topo, poussez dans A et lisez dans C

Etablir un lien bidirectionnel et lisez ecrivez dans A et B



Mettez en oeuvre un Producer et 2 Consumers

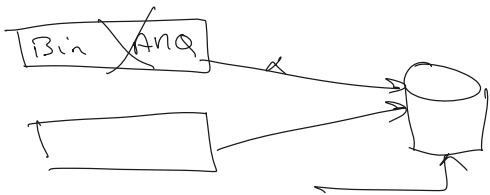
Pour permettre le maillage avec de multiples relais :
networkTTL="5" messageTTL="5" consumerTTL="5"
dans l'elements NetworkConnector

KahaDB

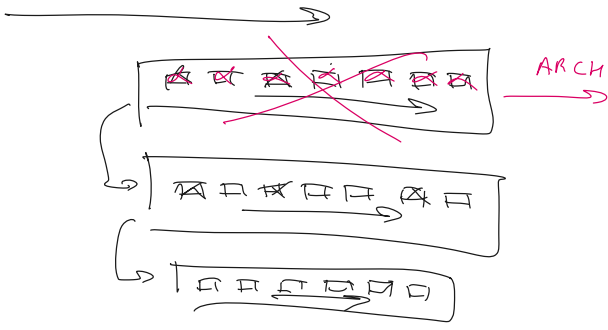
bin

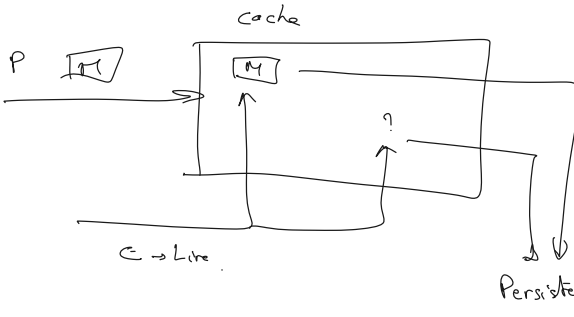
Storage (point de Montage → SAN)

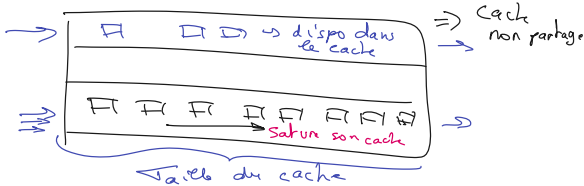
BDD JDBC



Messages



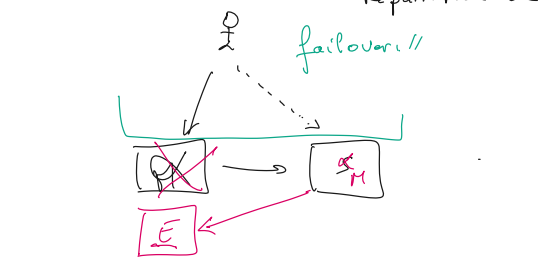


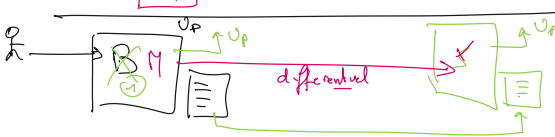


High Availability = Tolérance de panne

+

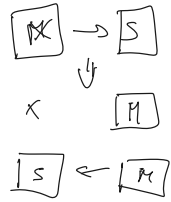
Répartition de charge





Exercice HA - shared-nothing:

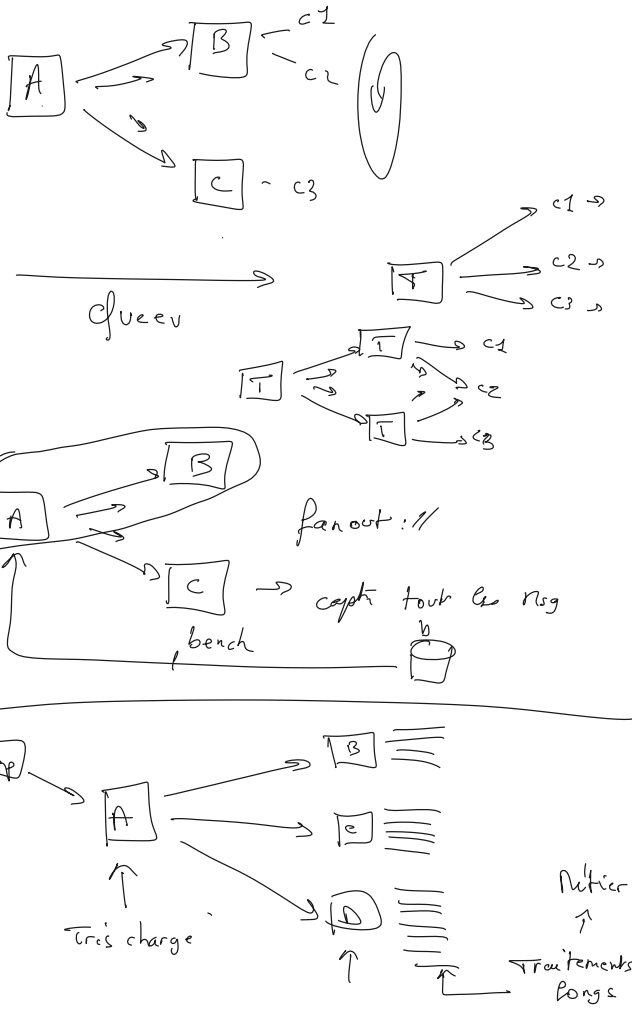
- Monter une infra :
  - Slave auto shutdown
- pousser des messages
  - Couper le Master
- Basculer le slave en Master
- Reconstruire l'ancien Master en new slave avec Bascule automatique
- Faire tomber le Master, le client doit pouvoir continuer à pousser les messages.

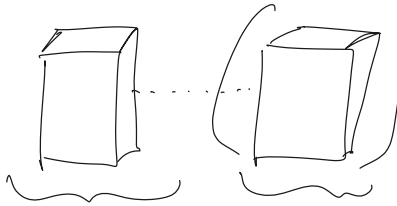


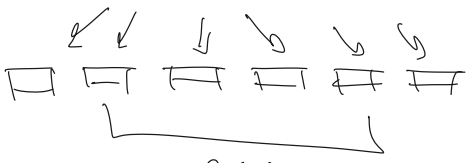
Avec un seul Broker,

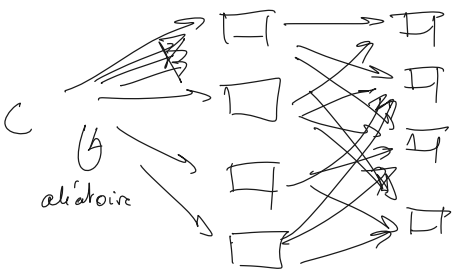
- Augmentez la taille mémoire consommée totale, et dont pour le système.
- passez en connections nio
- activez l'optimisation Dispatch

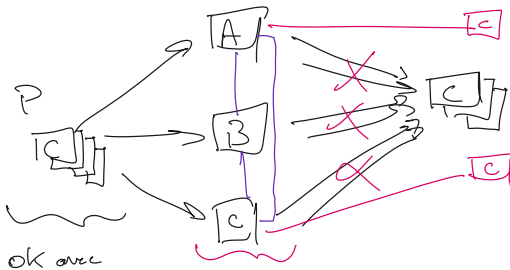
avec top et si possible JConsole, consultez la différence


A
B
C
c1
c2
c3
queeu
T
T
T
T
c1
c2
c3
fanout : //
bench
Appli
Tris charge
Multicr
Traitements
longs


Prod, gros
serveur,
cher, ...
x2

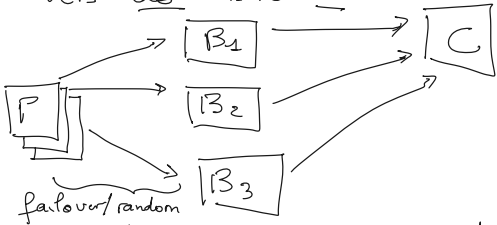

Réplication x3
→ 2 srv down


C
aléatoire



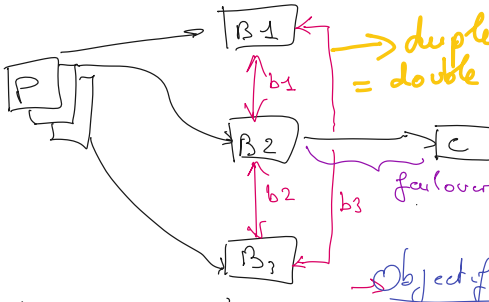
Mettre en oeuvre la répartition de charge

Objectif: Avoir 1..n Producers qui envoie (nt) des messages vers des Brokers



1:] Chaque Consumer établit n Connections vers les Brokers (pas mal de code à mettre en oeuvre)

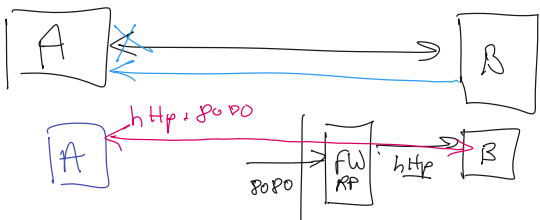
Second scénario :

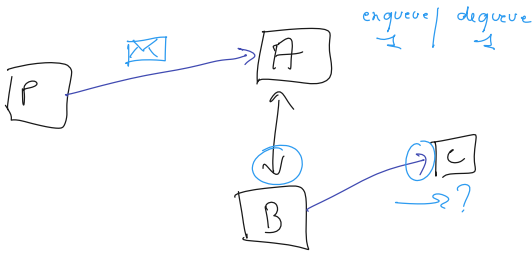


duplex = "true"
= double sens -

Objectif : n Producers / 1 Consumer, 3 bridges pour ne perdre aucun message

Tech : Lancer un Producer sur un autre broker que celui sur lequel notre Consumer est connecté :





Operations en ligne de commande

(A)

→ avec activemq --help

activemq purge --help,

Purgez une File

(B)

avec activemq producer --help

et activemq consumer --help,

sans passer par le projet Java,

lancez un consumer,

puis un producer sur une file

Modifiez : - le contenu du message texte

- le nombre de messages

Tip: benchmark possible avec time <commande>

ex: time activemq producer ...

essai tcp et nio

Architecture de Haute dispo (Tolérance + répartition)