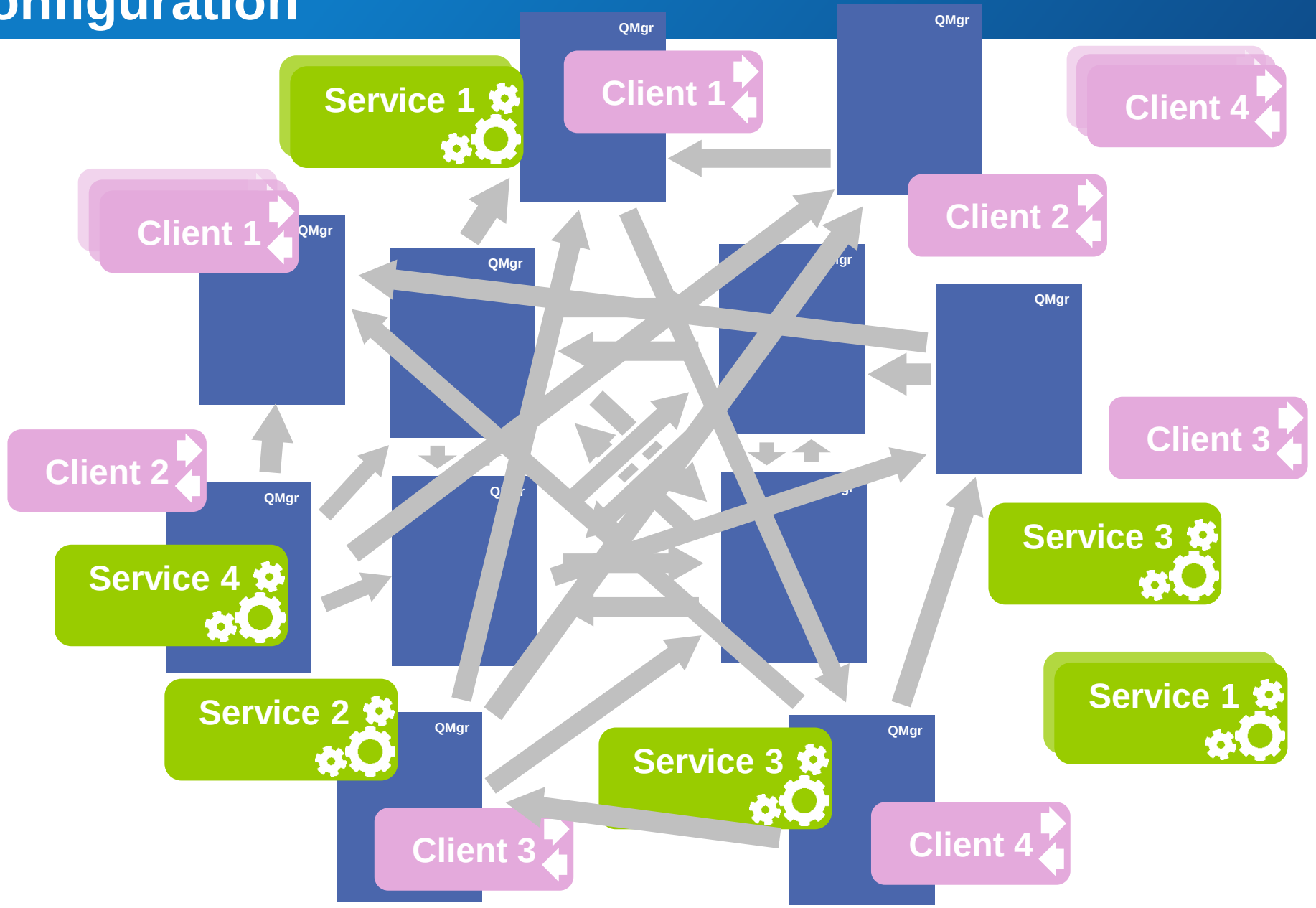
Recording available at:
http://www.slideshare.net/DavidWare1/ame-2273-mq-clustering-pdf

What can you achieve with MQ clusters?

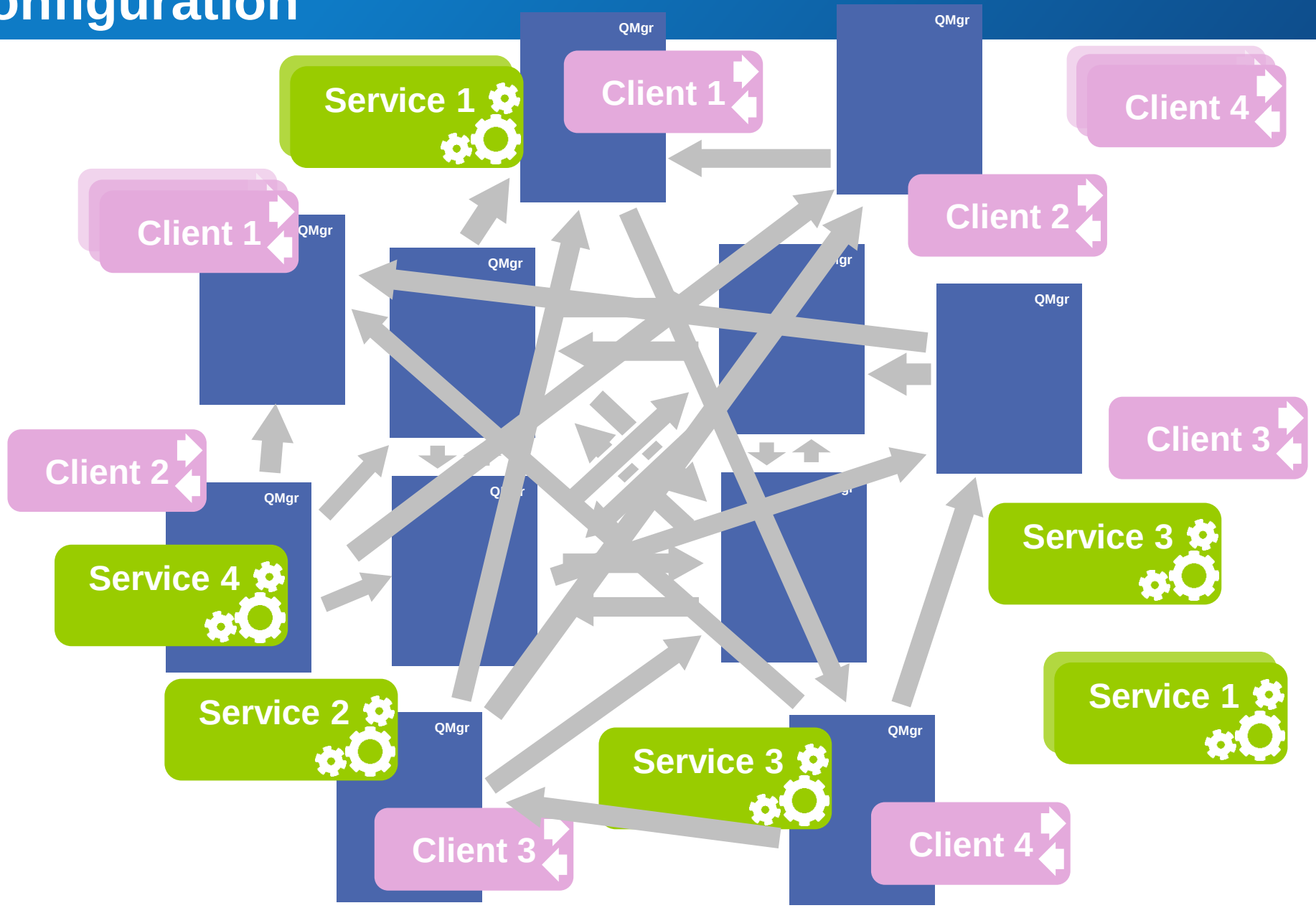
David Ware
Lead Architect, IBM MQ Distributed
dware@uk.ibm.com

Scaling your administration

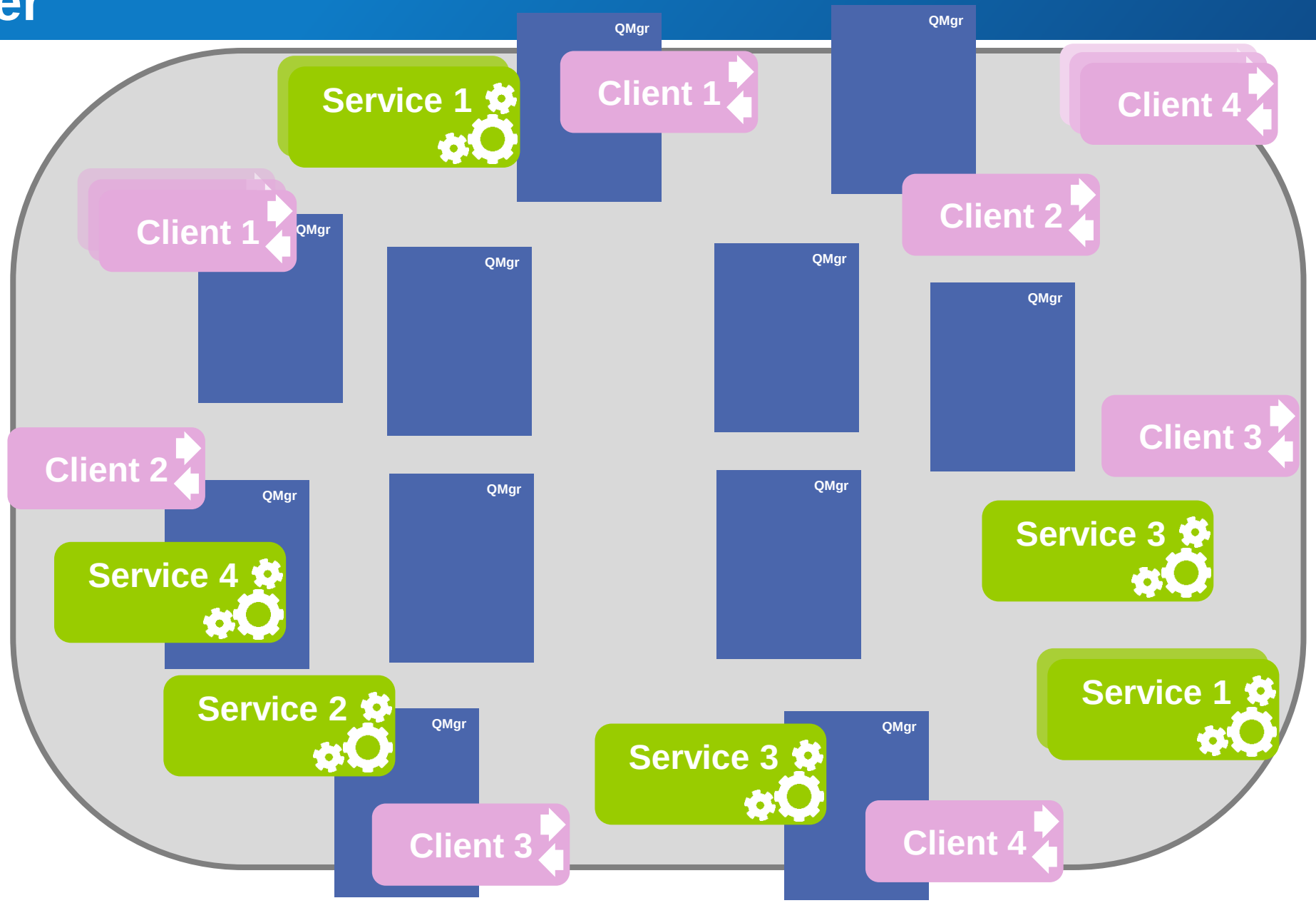
Manual configuration



Manual configuration

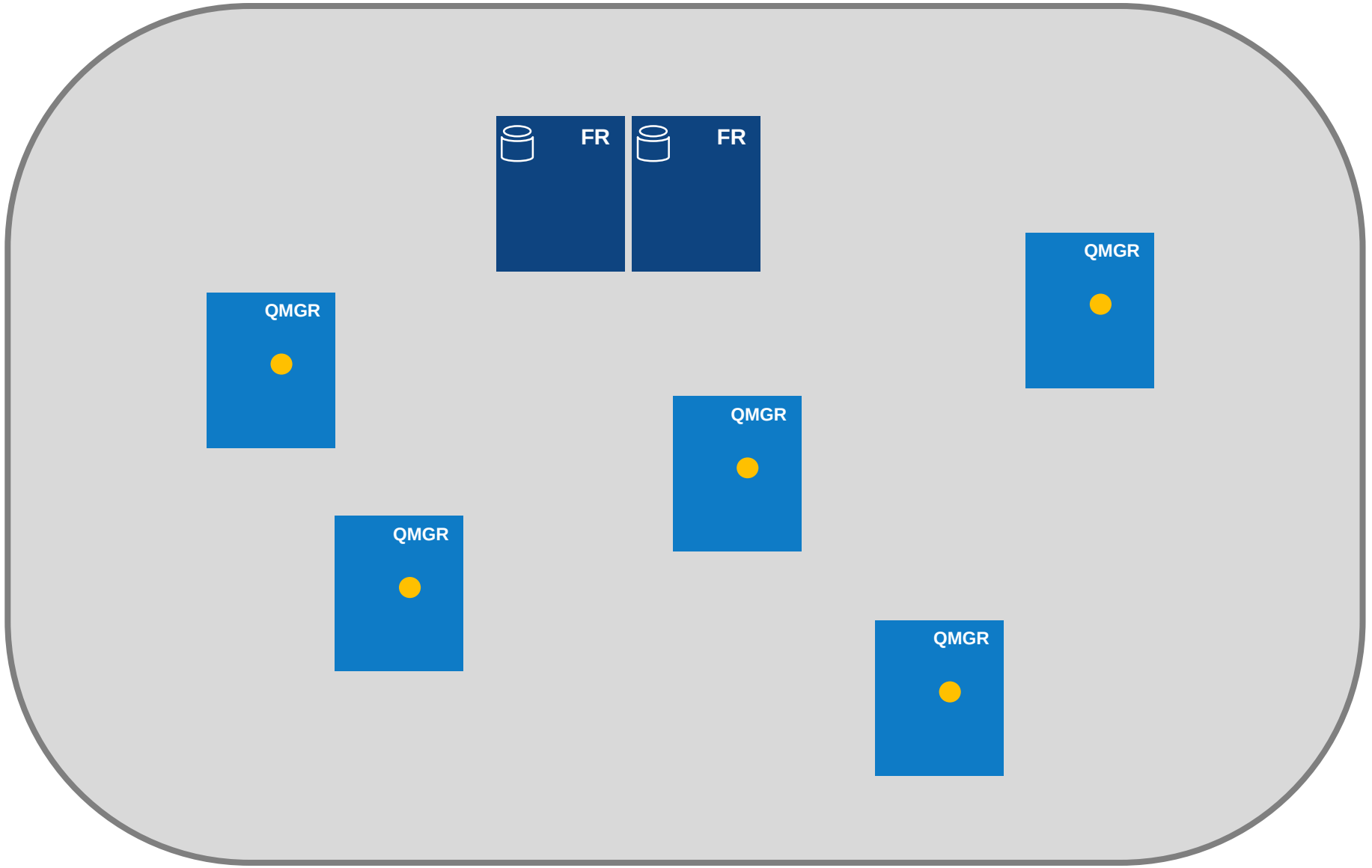


MQ Cluster

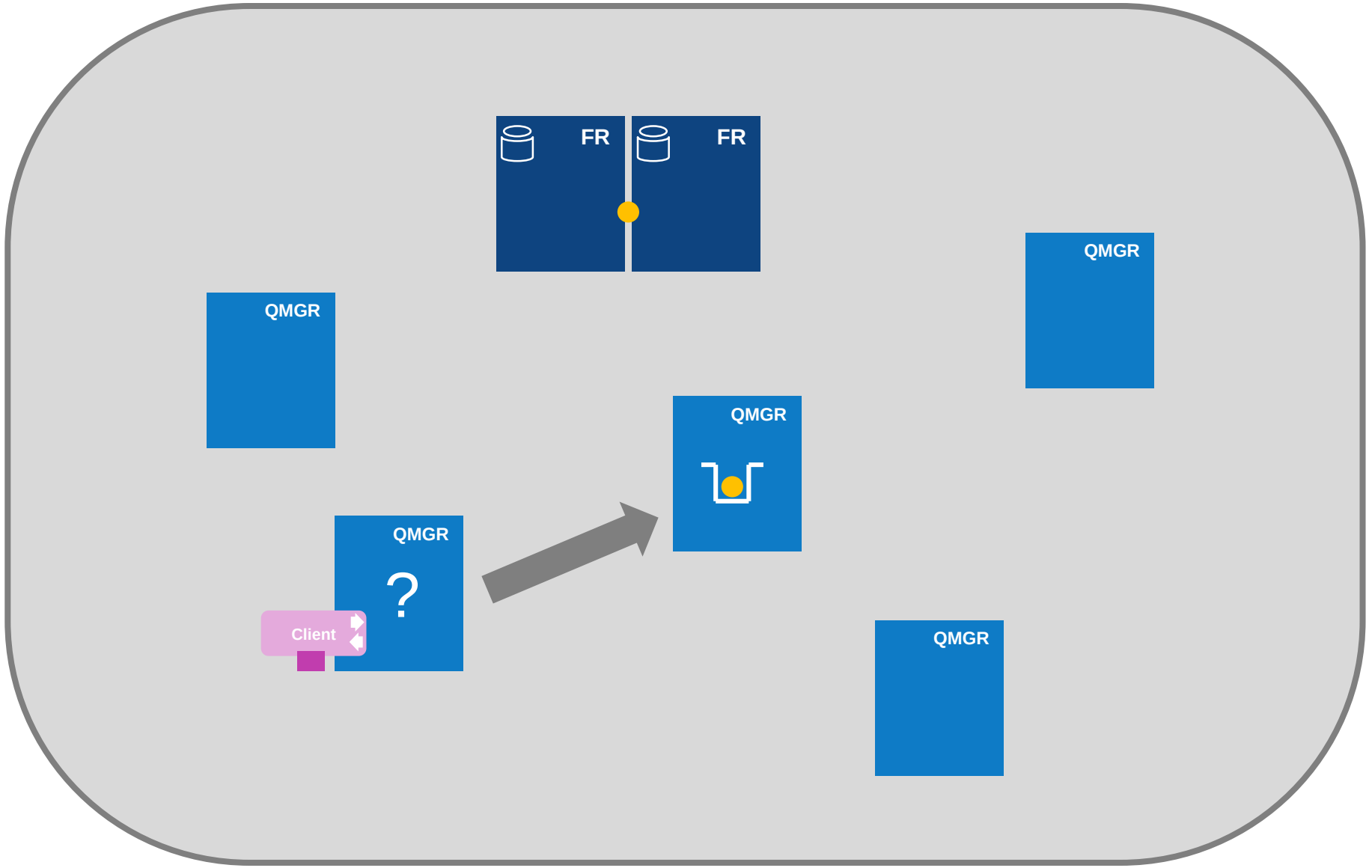


How it works

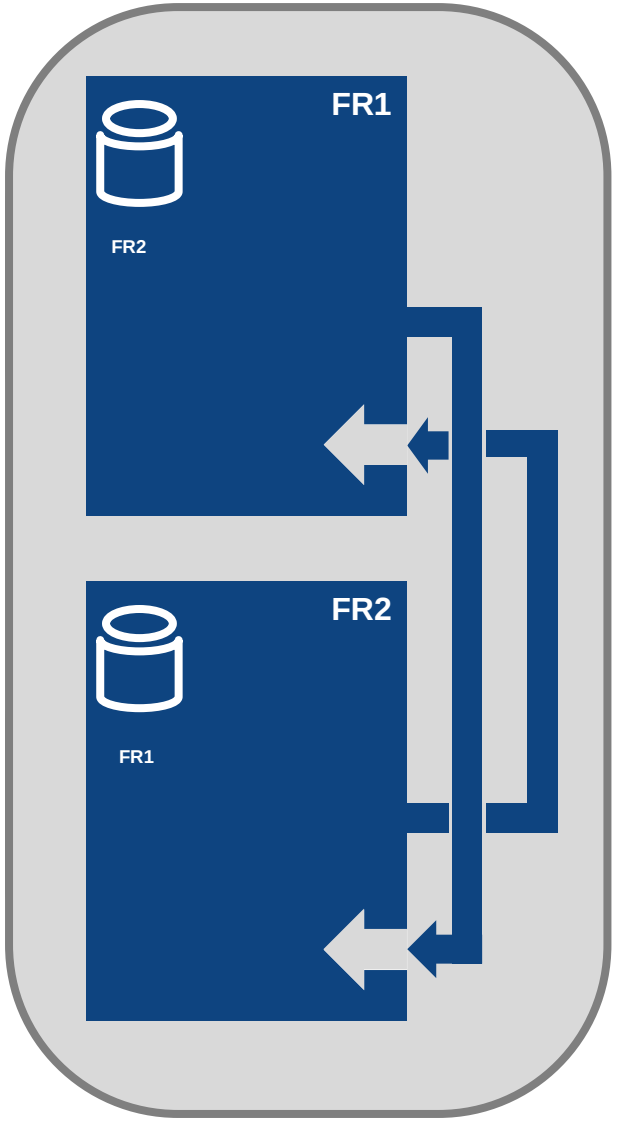
Building a cluster



Building a cluster



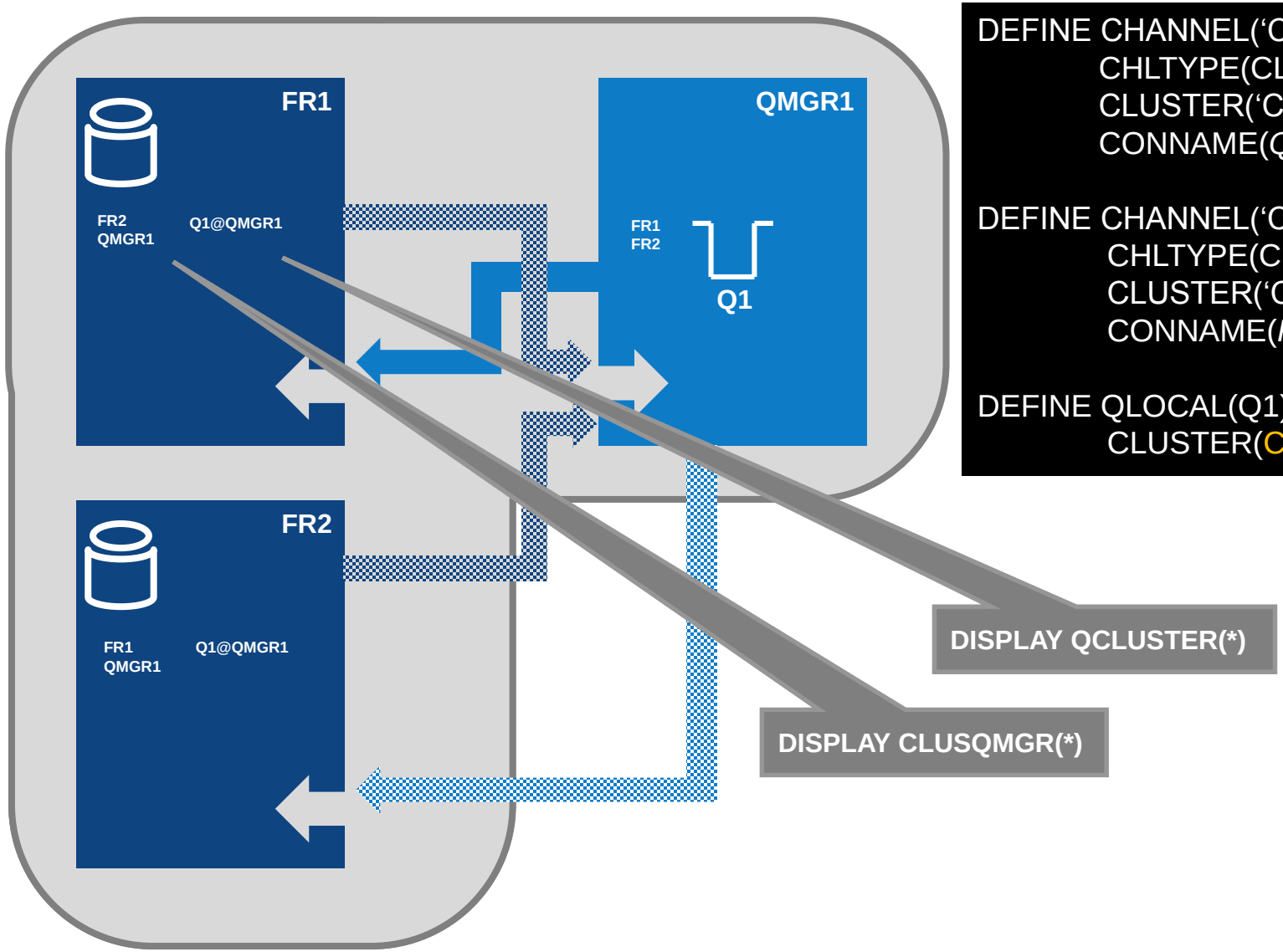
Step 1: Create your two full repositories




```
ALTER QMGR REPOS('CLUS1')

DEFINE CHANNEL('CLUS1.FR1') CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1') CONNAME(FR1 location)

DEFINE CHANNEL('CLUS1.FR2') CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1') CONNAME(FR2 location)
```

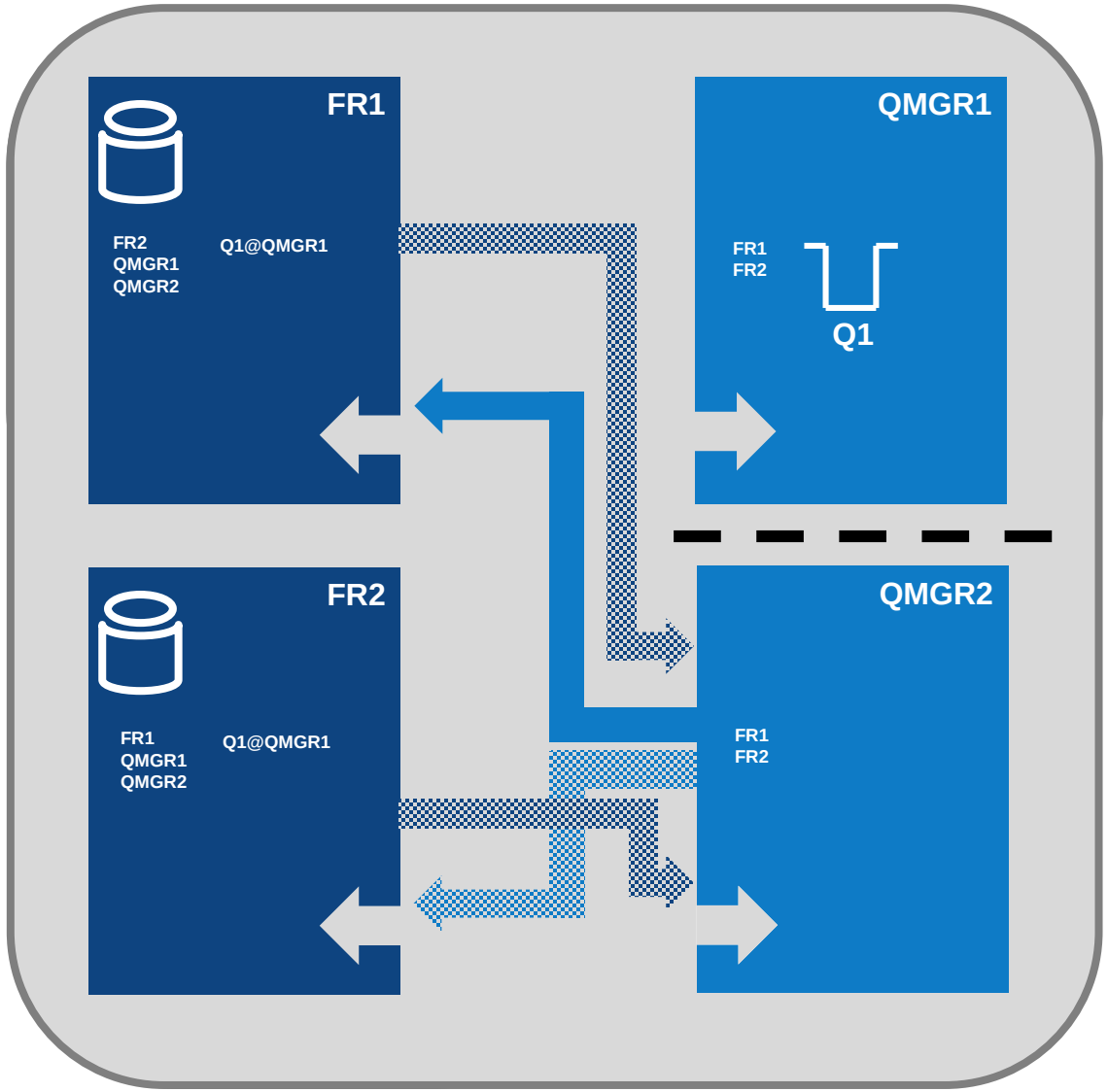
```
ALTER QMGR REPOS('CLUS1')

DEFINE CHANNEL('CLUS1.FR2') CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1') CONNAME(FR2 location)

DEFINE CHANNEL('CLUS1.FR1') CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1') CONNAME(FR1 location)
```

Step 2: Add in more queue managers




```
DEFINE CHANNEL('CLUS1.QMGR1')
       CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1')
       CONNAME(QMGR1 location)

DEFINE CHANNEL('CLUS1.FR1')
       CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1')
       CONNAME(FR1 location)

DEFINE QLOCAL(Q1)
       CLUSTER(CLUS1)
```

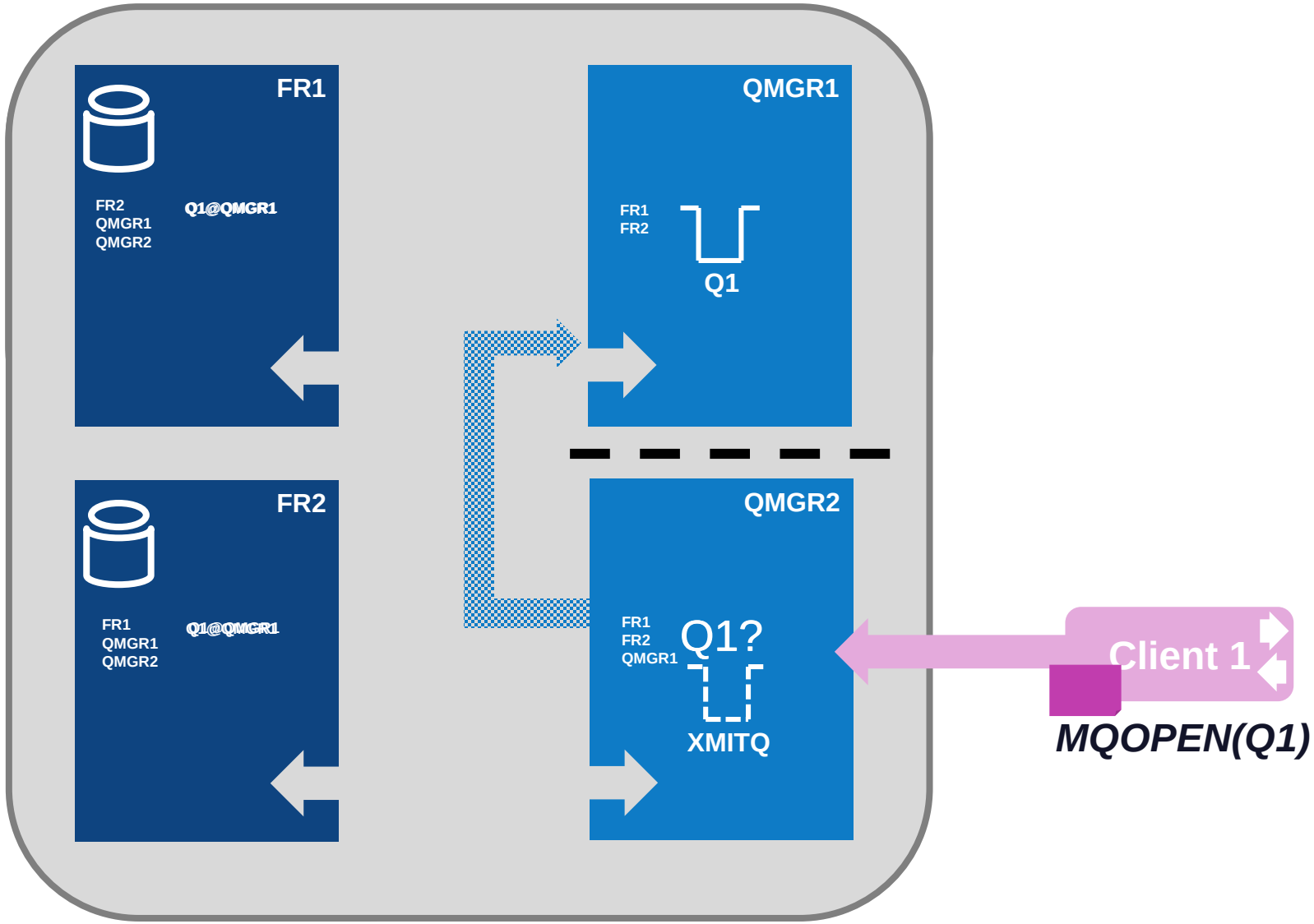
Step 2: Add in more queue managers



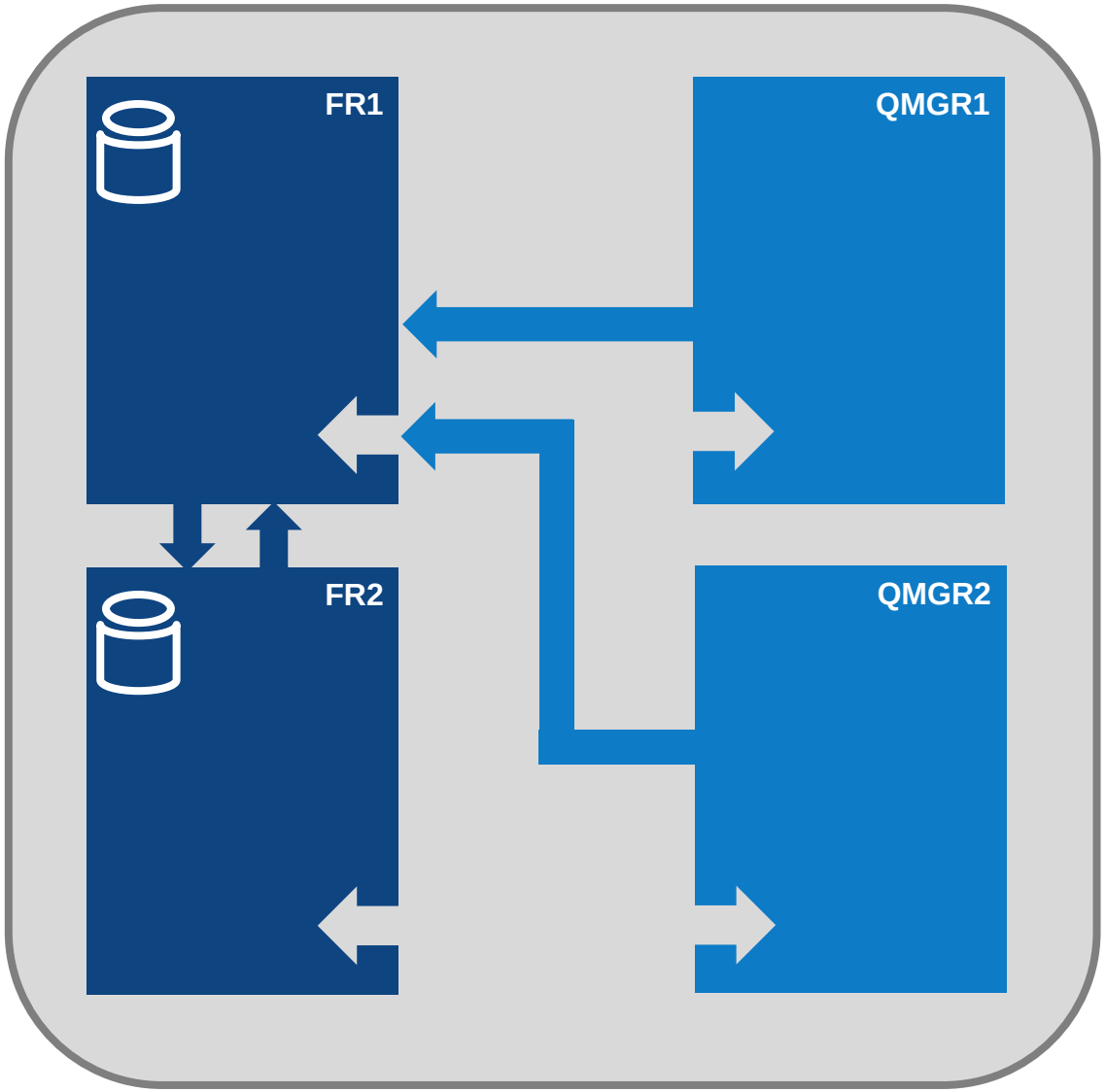

```
DEFINE CHANNEL('CLUS1.QMGR1')
       CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1')
       CONNAME(QMGR1 location)

DEFINE CHANNEL('CLUS1.FR1')
       CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1')
       CONNAME(FR1 location)

DEFINE QLOCAL(Q1)
       CLUSTER(CLUS1)
```

```
DEFINE CHANNEL('CLUS1.QMGR2')
       CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1')
       CONNAME(QMGR2 location)

DEFINE CHANNEL('CLUS1.FR1')
       CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1')
       CONNAME(FR1 location)
```

Step 3: Start sending messages



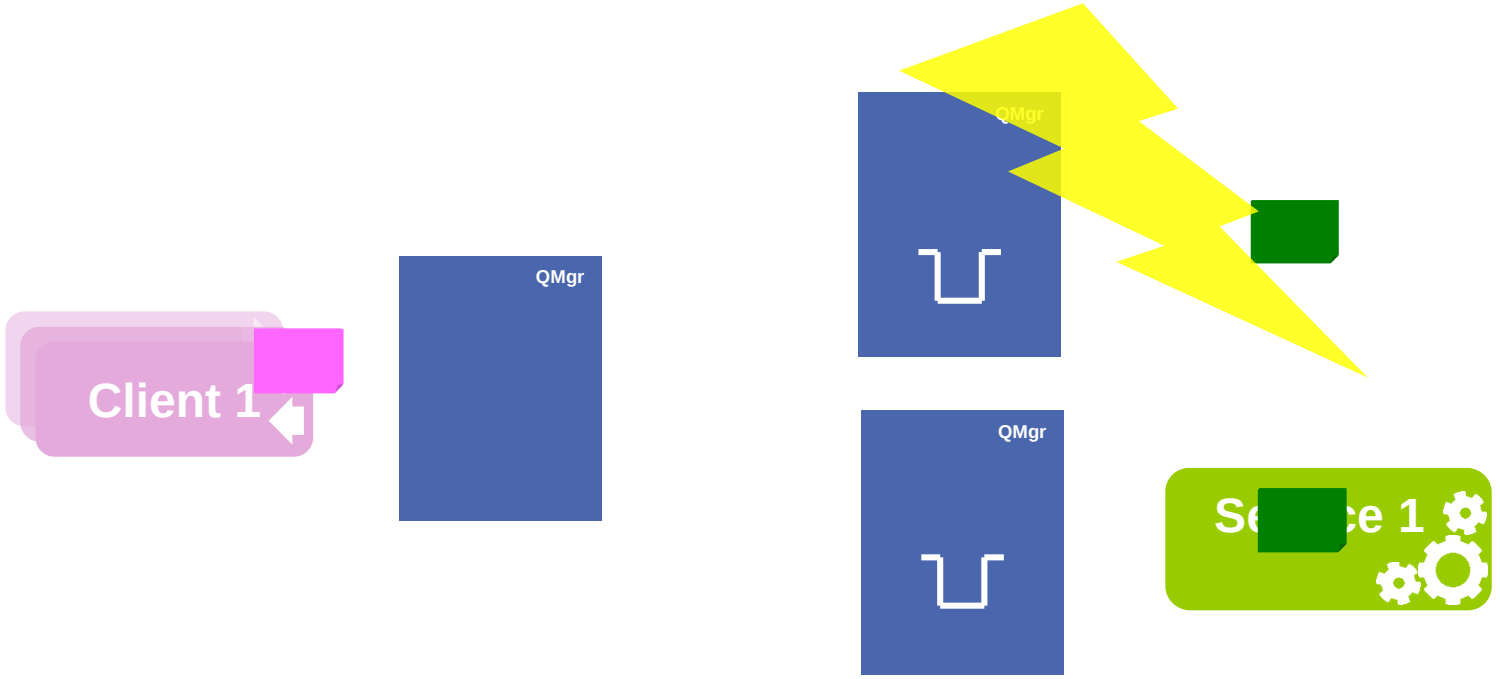
So all you needed…



- Two full repository queue managers
- A cluster receiver channel each
- A single cluster sender each
- No need to manage pairs of channels between each queue manager combination or their transmission queues
- No manual starting of channels
- No need for remote queue definitions or transmission queues

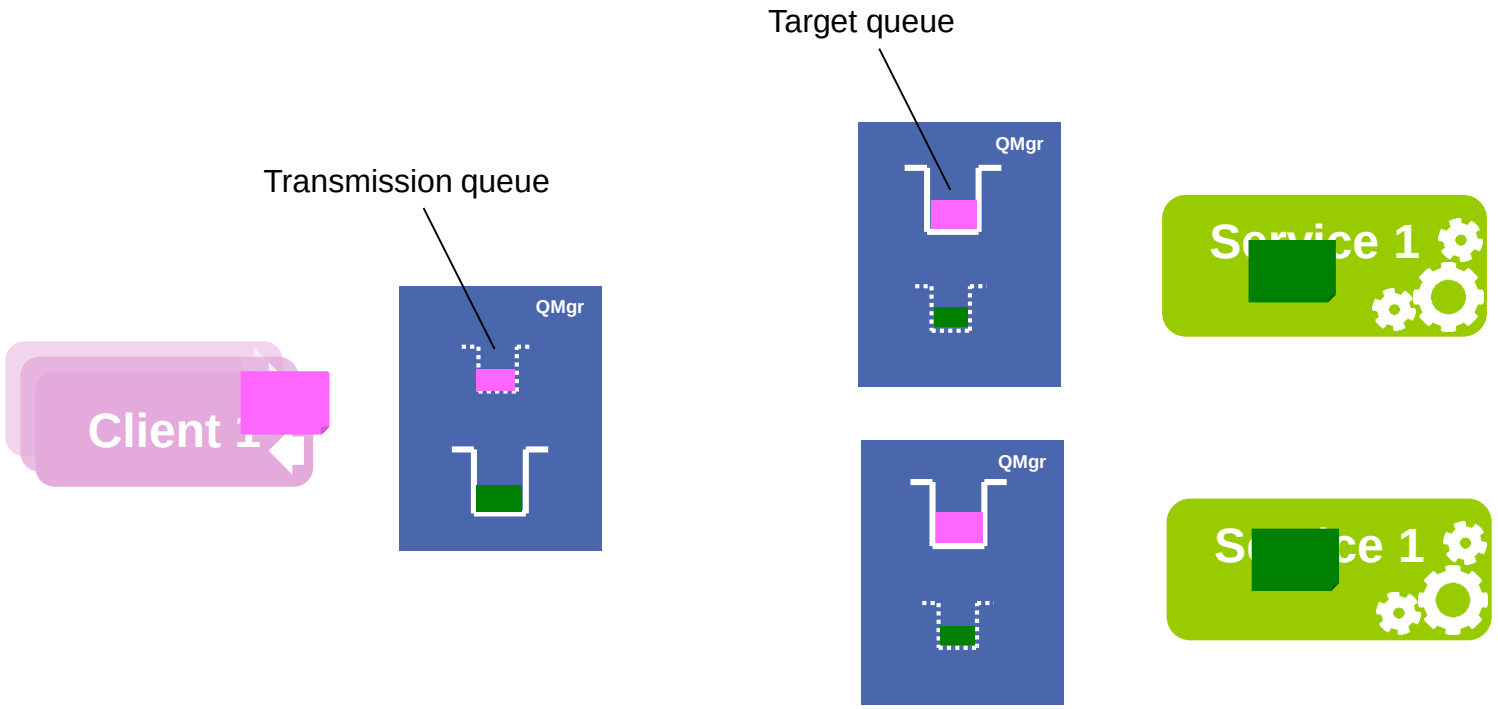
But that's just for starters…

But what else do you get with a cluster?



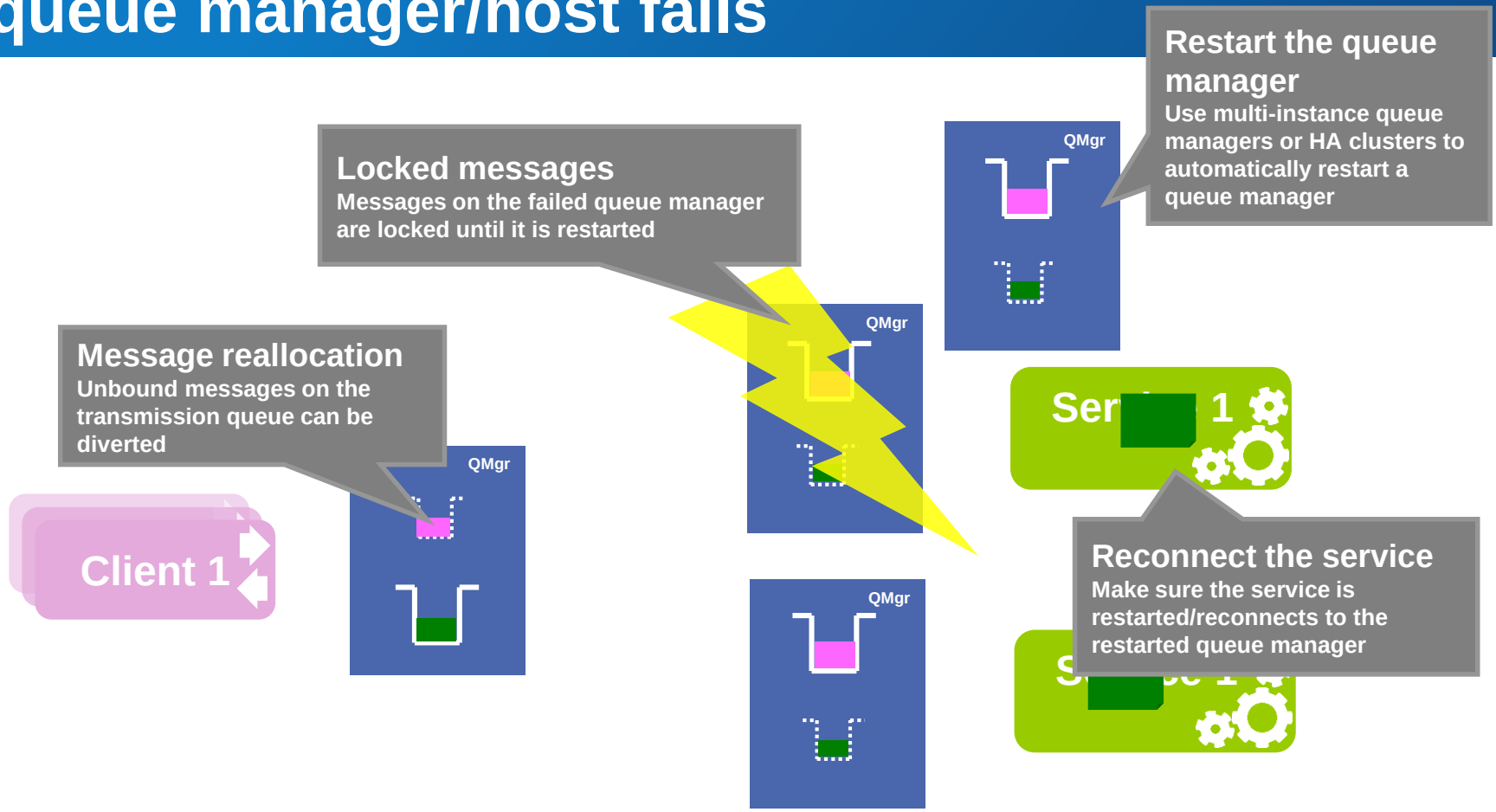
- Workload Balancing
- Service Availability

Where can the messages get stuck?



- Target queues
- Transmission queues

The service queue manager/host fails

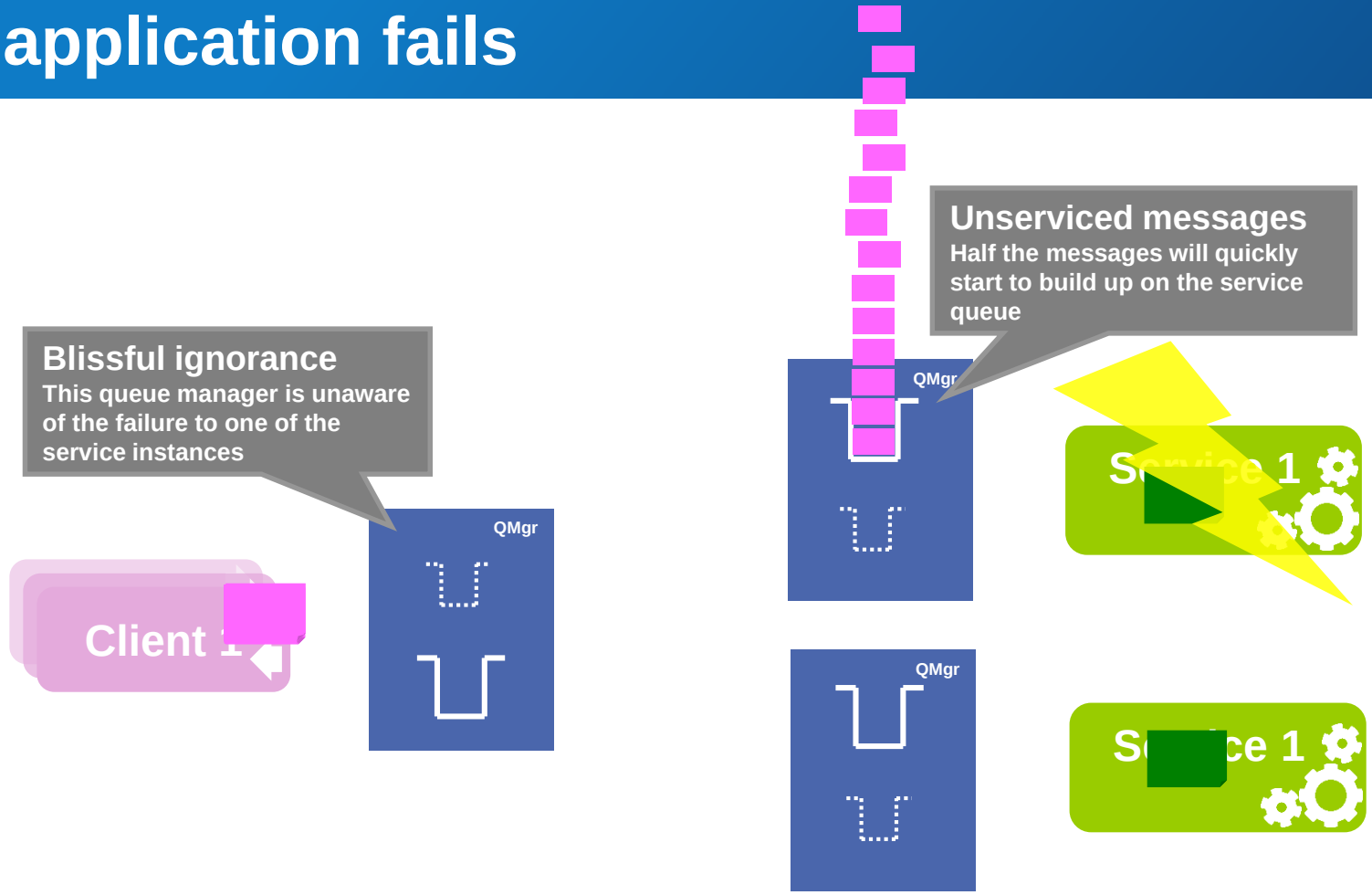


When a queue manager fails:

- Ensure messages are not **bound** to it
- Restart it to release queued messages

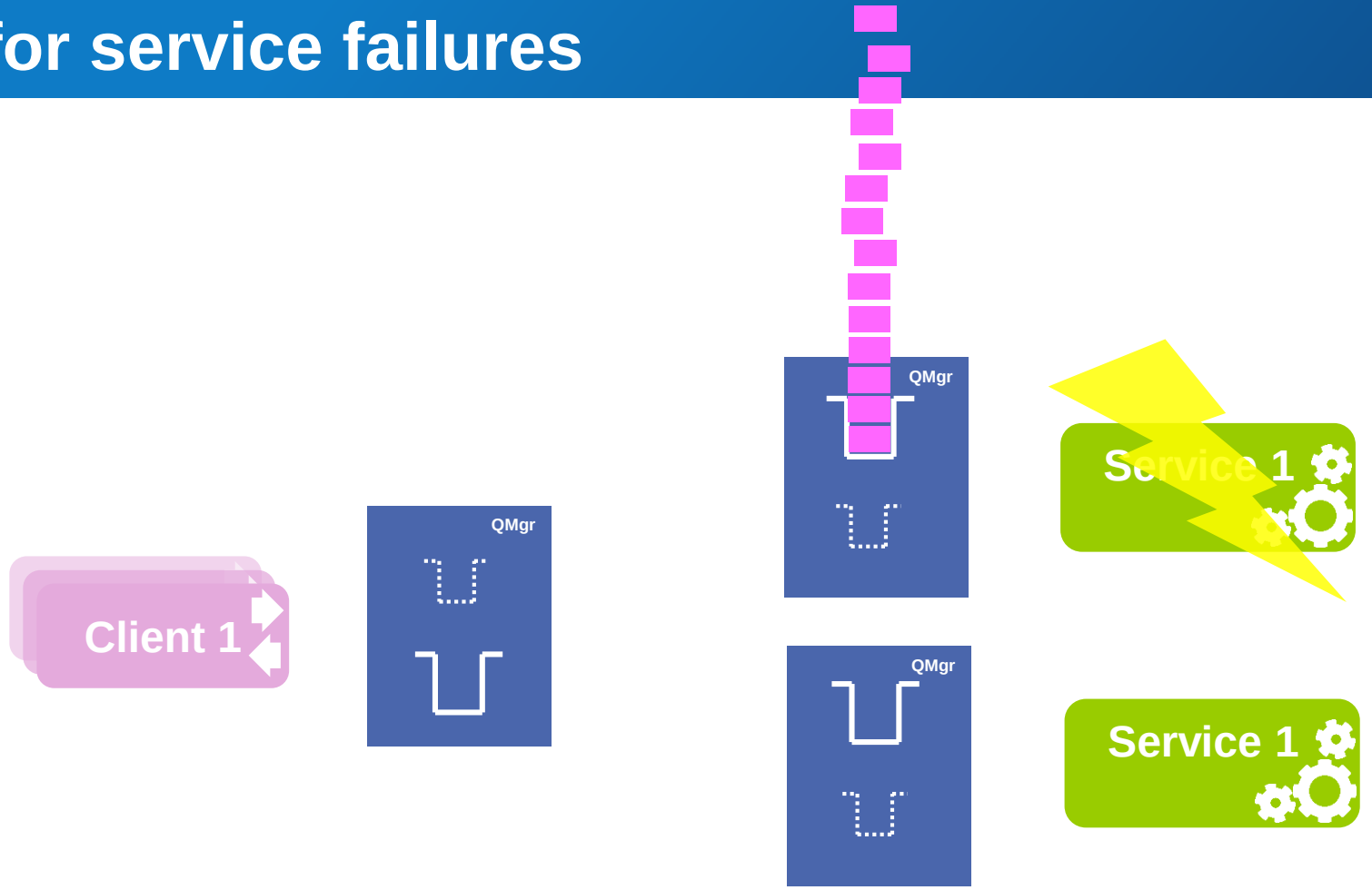
Service application availability

The service application fails

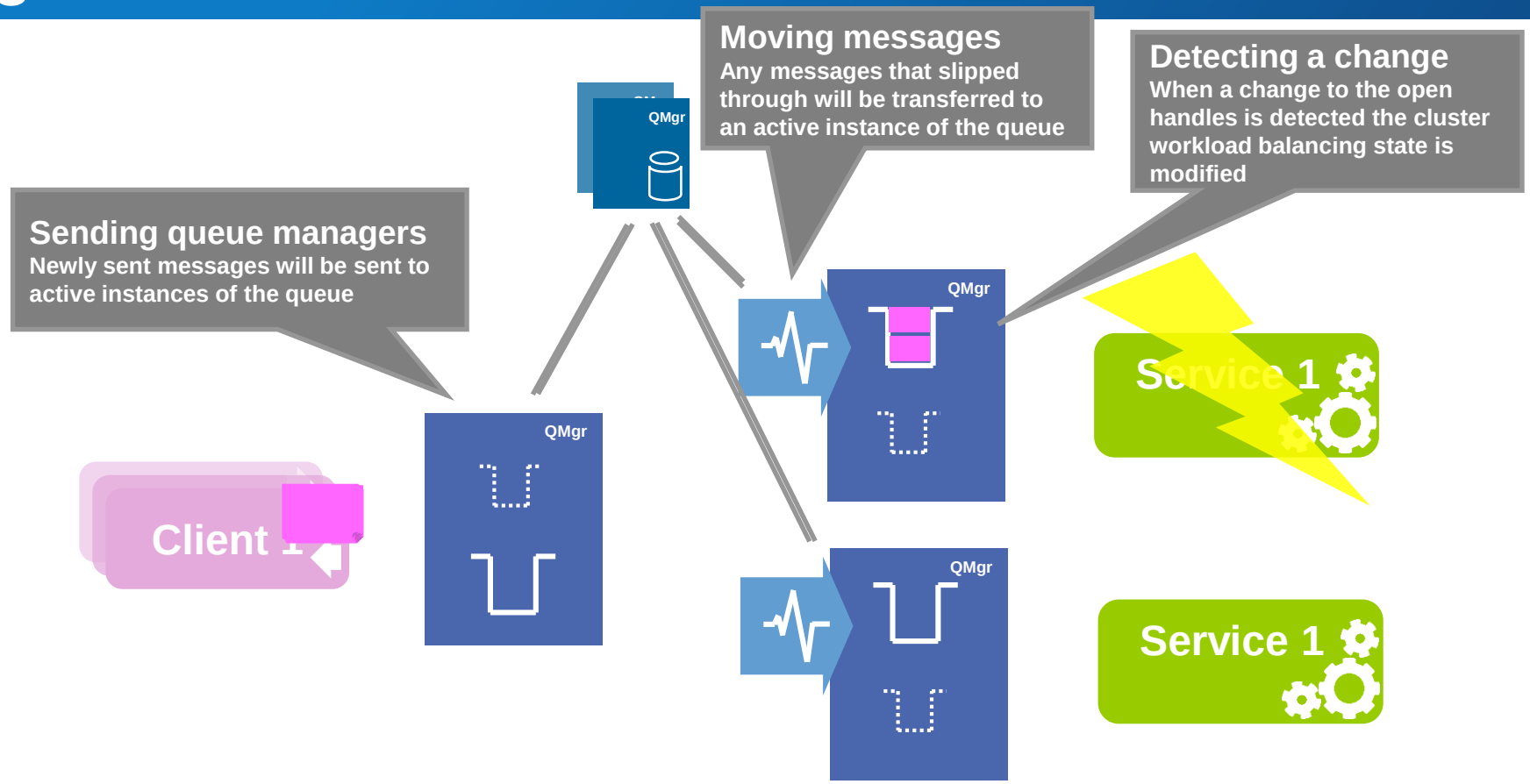


- Cluster workload balancing does not take into account the availability of receiving applications.
- Or a build up of messages.

Monitoring for service failures



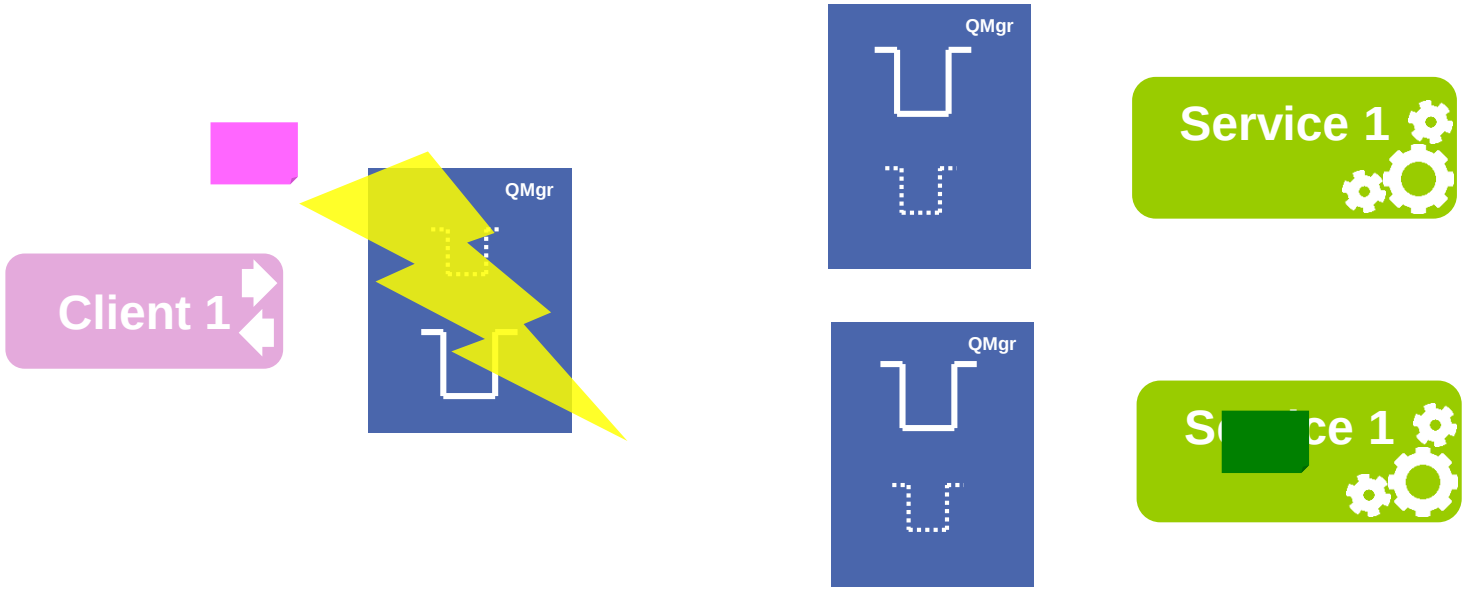
Monitoring for service failures



- MQ provides a sample monitoring service
- Regularly checks for attached consuming applications
- Generally suited to ***steady state*** service applications

Client failures

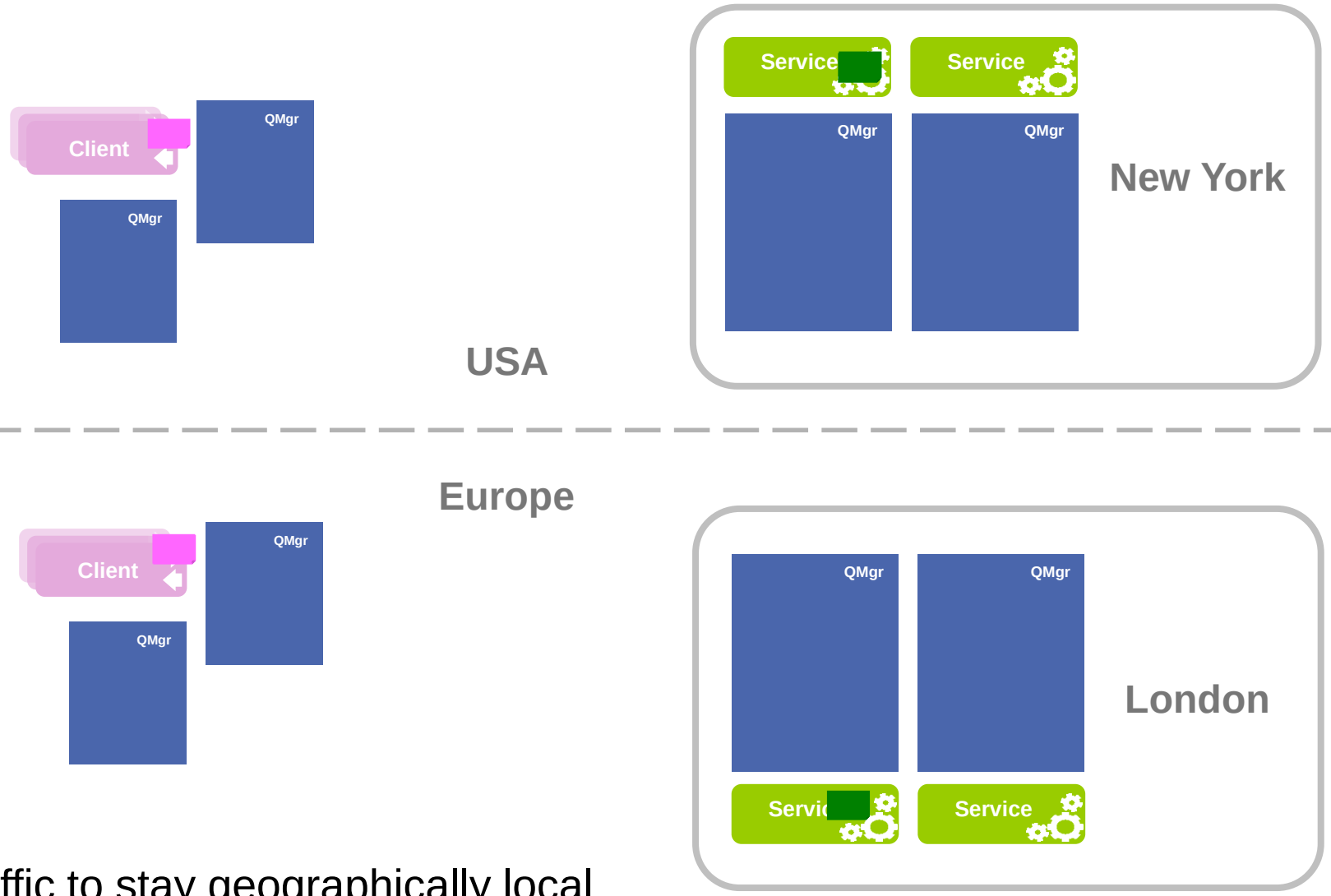
Client availability



- Multiple locations for a client to connect to
  - Allows new requests when one queue manager is unavailable.
- Replies can be automatically routed back to the originating queue manager.

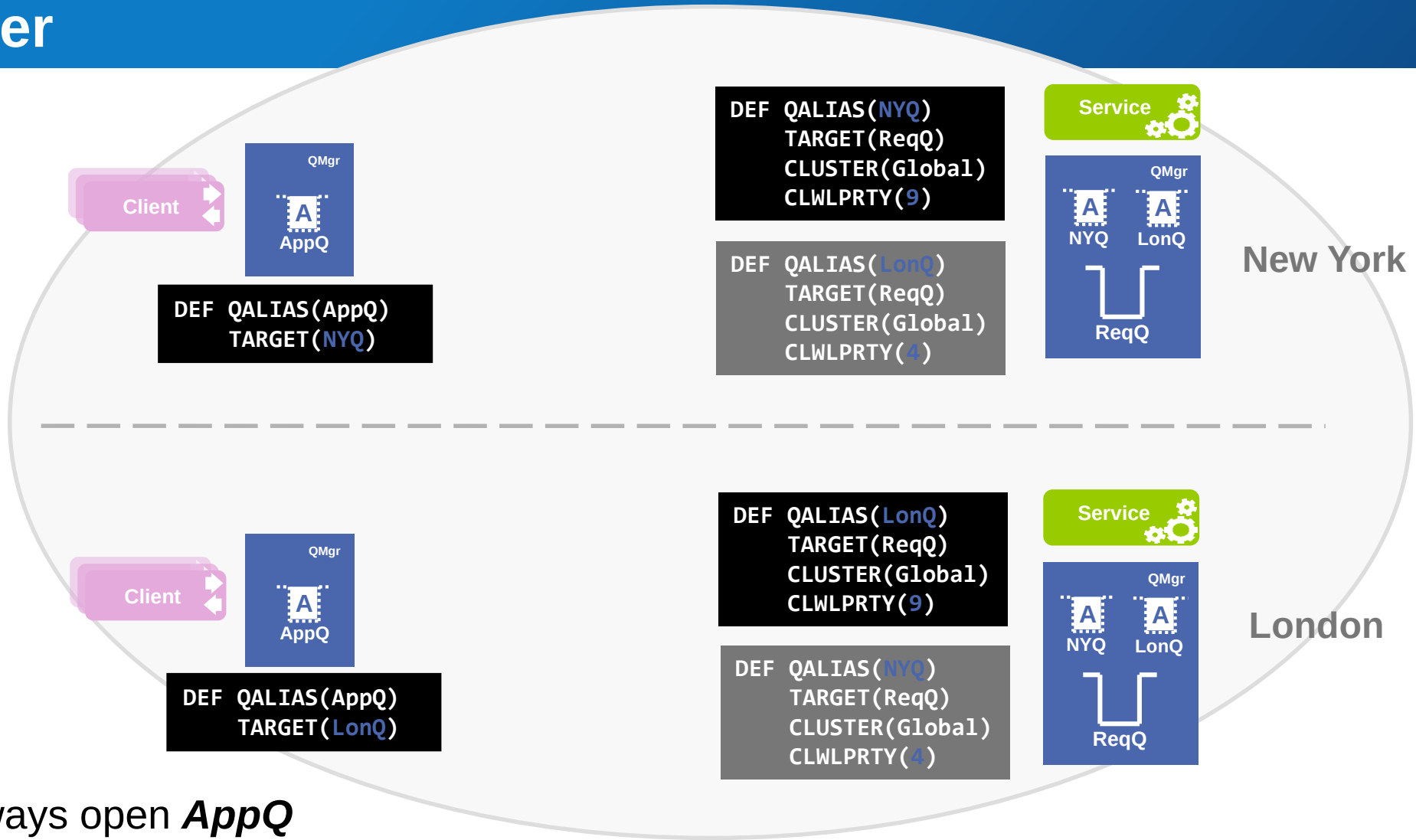
Smarter routing

Global applications



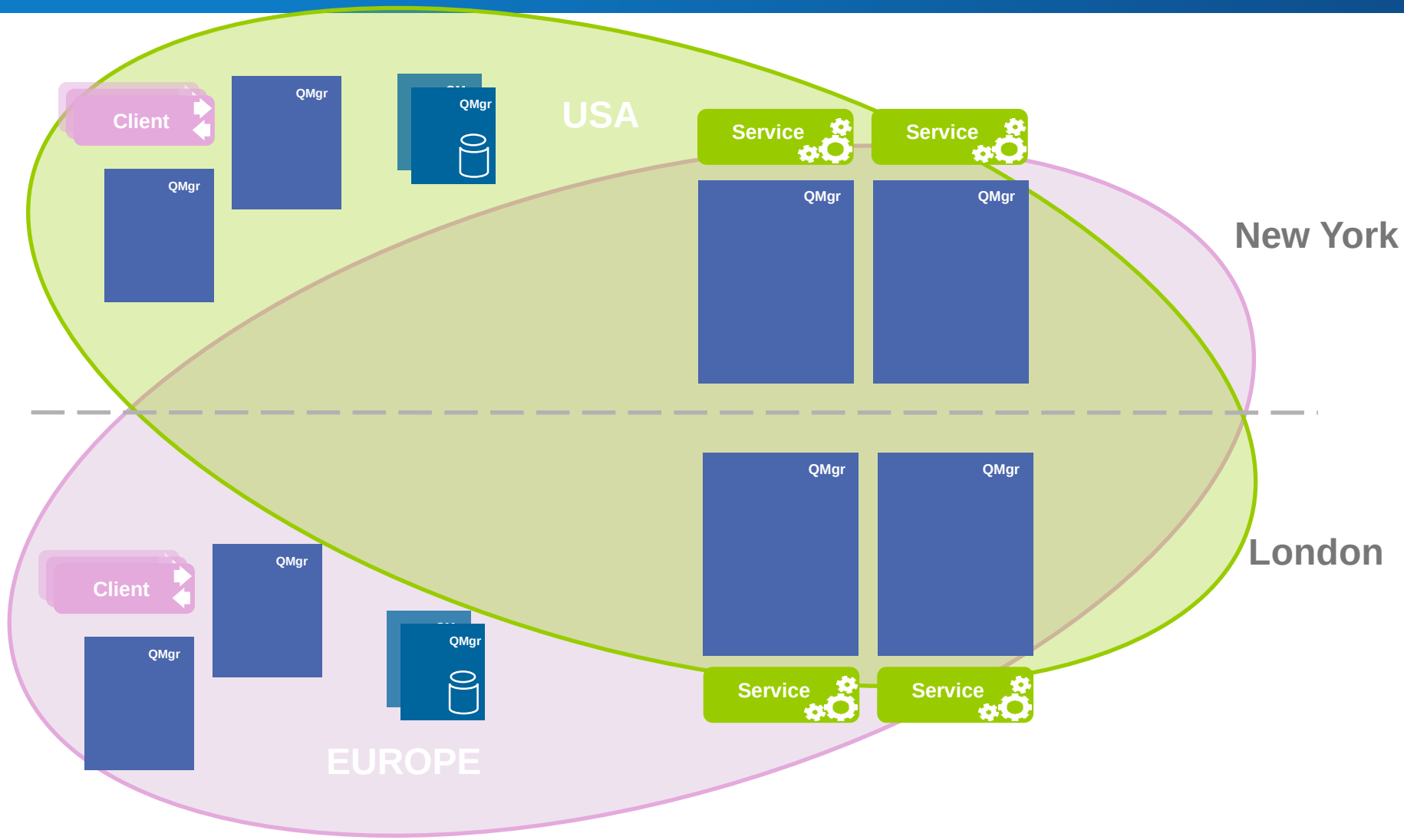
- Prefer traffic to stay geographically local
- Except when you have to look further afield
- How do you achieve this with clusters?

One cluster



- Clients always open ***AppQ***
- Local alias determines the preferred region
- Cluster workload priority is used to target geographically local cluster aliases
- Use of CLWLPRTY enables automatic failover
  - CLWLRANK can be used for manual failover

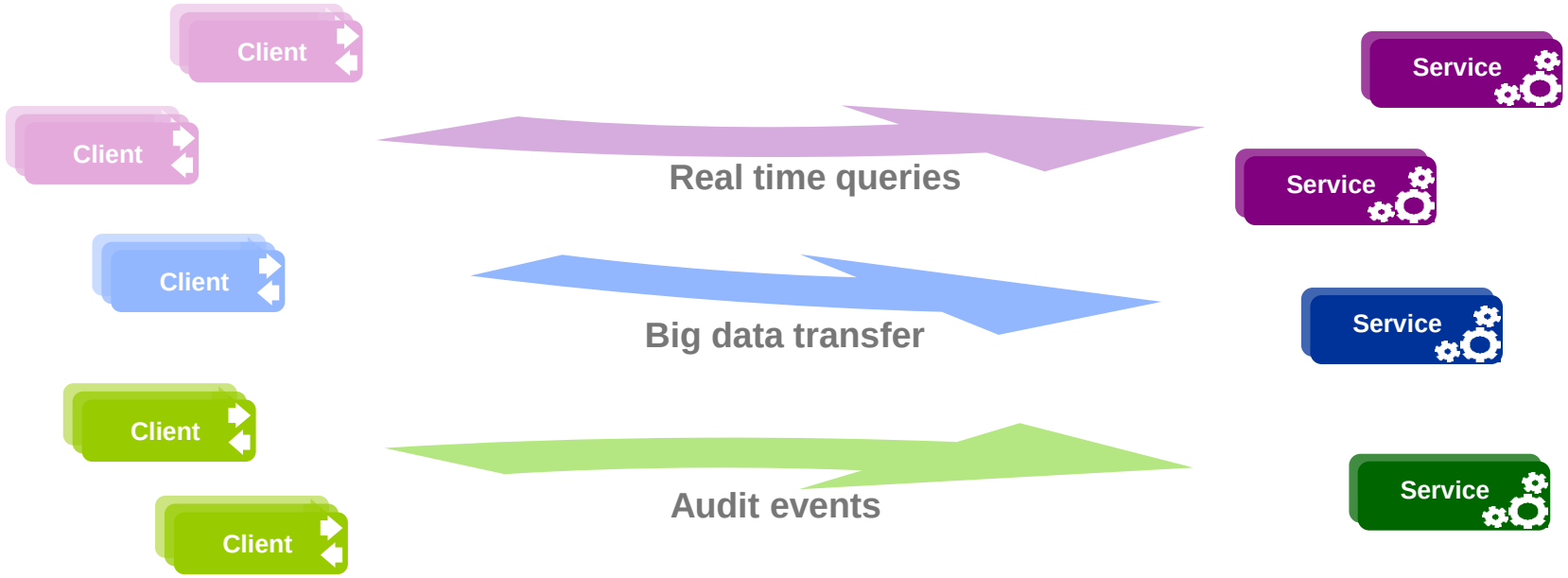
The two cluster alternative



- The ***service*** queue managers join ***both*** geographical clusters
  - Each with separate cluster receivers for each cluster, at ***different cluster priorities***. Queues are clustered in ***both*** clusters.
- The ***client*** queue managers are in their ***local*** cluster only.

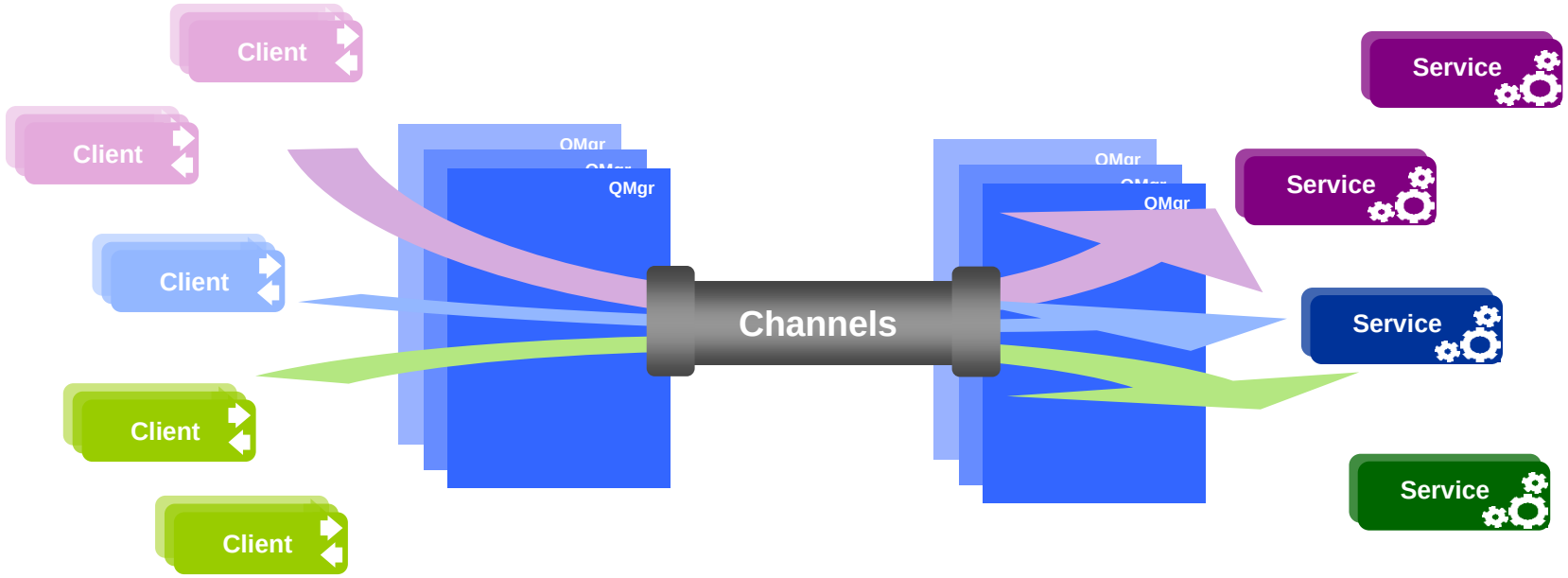
Separation of traffic

Multiple types of traffic



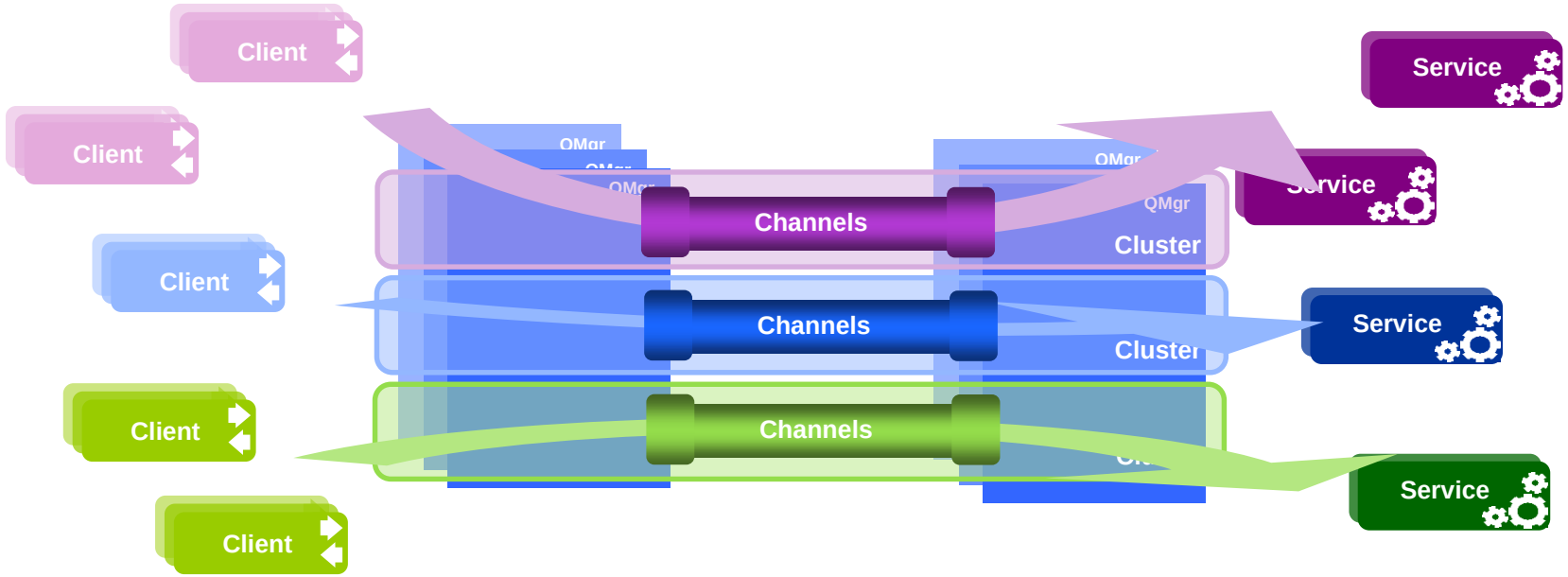
- Often a IBM MQ backbone will be used for multiple types of traffic

Multiple types of traffic



- Often a IBM MQ backbone will be used for multiple types of traffic
- When using a single cluster and the same queue managers, messages all share the same channels
- Even multiple cluster receiver channels in the same cluster will not separate out the different traffic types

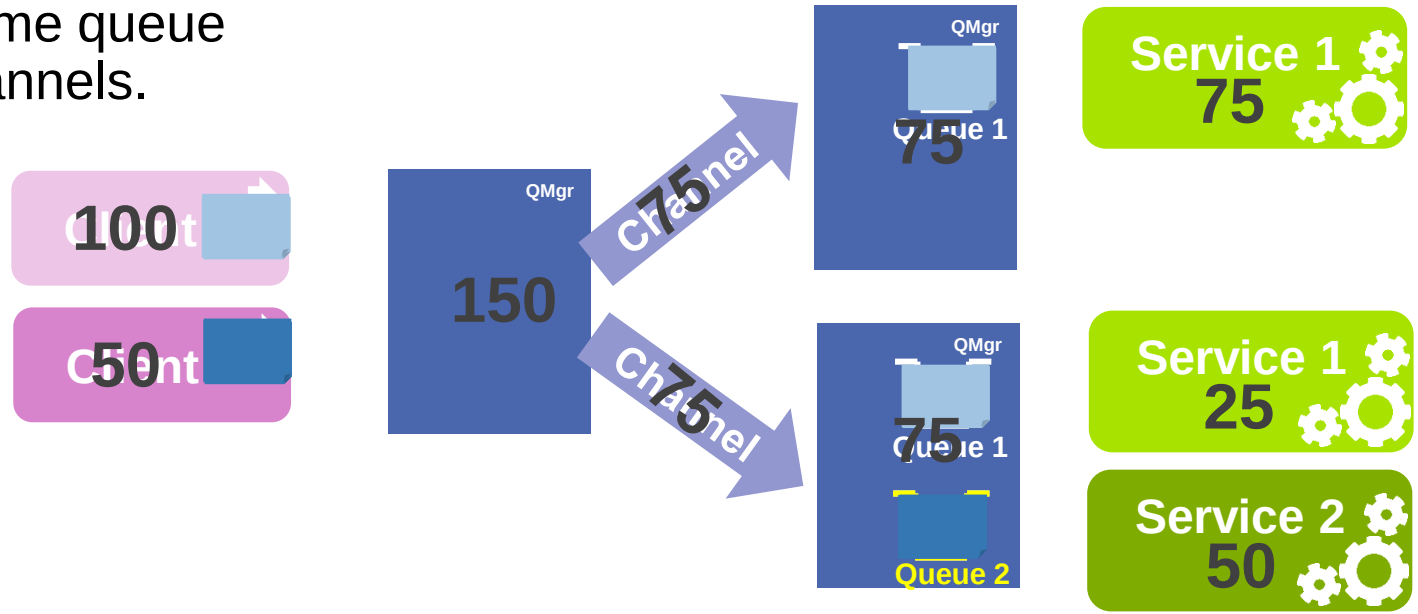
Multiple types of traffic



- Often a IBM MQ backbone will be used for multiple types of traffic
- When using a single cluster and the same queue managers, messages all share the same channels
- Even multiple cluster receiver channels in the same cluster will not separate out the different traffic types
- Multiple overlaid clusters with different channels enable separation

Workload balancing level interference

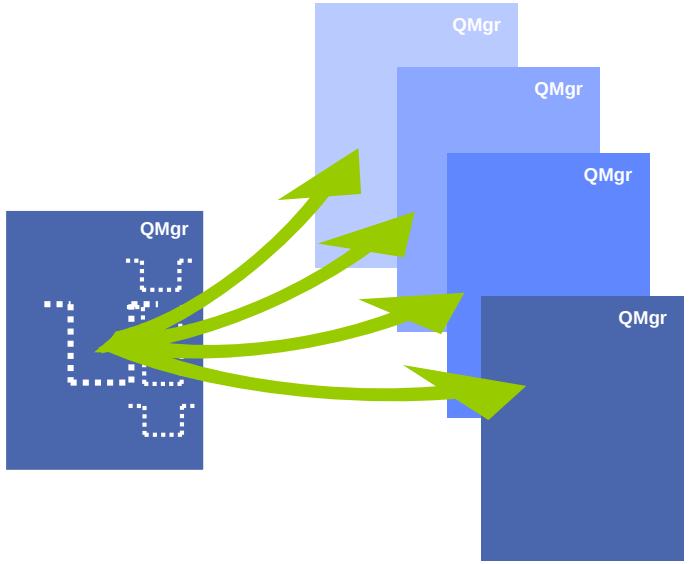
- Multiple applications sharing the same queue managers and the same cluster channels.



- Cluster workload balancing is at the **channel** level.
  - Messages sharing the same channels, but to different target queues will be counted together.
- The two channels here have an even 50/50 split of messages…
- *…but the two instances of Service 1 do not!*
- *Split Service 1 and Service 2 queues out into separate clusters, queue managers or customise workload balancing logic.*

Cluster transmit queues

- **The default is a single shared transmission queue for all of a queue manager's cluster traffic**
- **MQ V7.5/V8.0 added multiple cluster transmission queue support**



- **Separation of Message Traffic**
  - With a single transmission queue there is potential for pending messages for cluster *ChannelA* to interfere with messages pending for cluster *ChannelB*
- **Management of messages**
  - Use of queue concepts such as *MAXDEPTH* not useful when using a single transmission queue for more than one channel.
- **Monitoring**
  - Tracking the number of messages processed by a cluster channel currently difficult/impossible using queue.
- **Performance?**
  - In reality a shared transmission queue is not always the bottleneck, often other solutions to improving channel throughput *(e.g. multiple cluster receiver channels)* are really what's needed.

Summary

- Setting up a cluster
- Service availability
- Location dependency
- Avoiding interference

WHERE DO I GET MORE INFORMATION?

IBM Messaging developerWorks
developer.ibm.com/messaging
www.ibm.com/developerworks/community/blogs/messaging

IBM Messaging Youtube
https://www.youtube.com/IBMmessagingMedia

LinkedIn
Ibm.biz/ibmmessaging

Twitter
@IBMMessaging

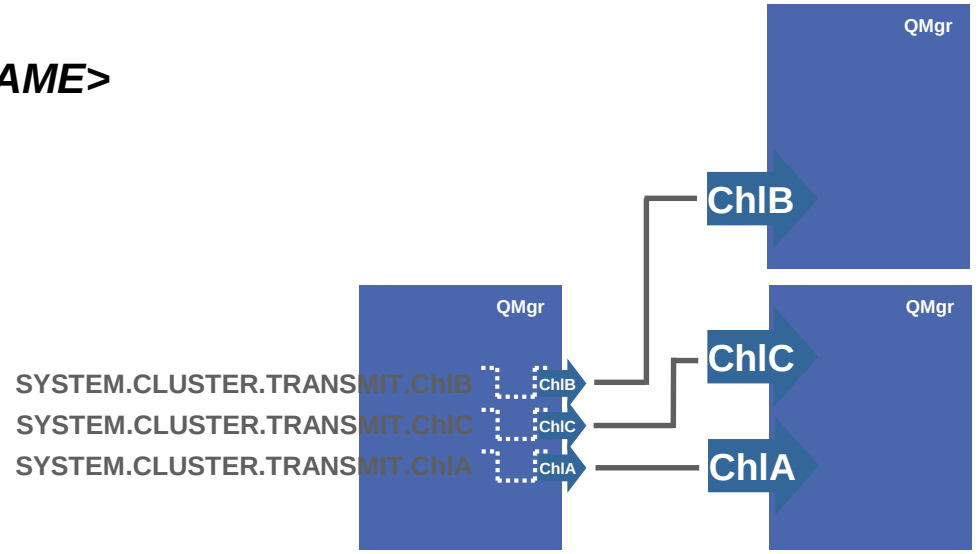
IBM MQ Facebook
Facebook.com/IBM-MQ-8304628654/

Multiple cluster transmit queues: Automatic

- **Configured on the *sending queue manager*, not the owners of the cluster receiver channel definitions.**

- **Queue Manager switch to automatically create a dynamic transmission queue per cluster sender channel.**
  ALTER QMGR DEFCLXQ(CHANNEL)
- **Dynamic queues based upon model queue. SYSTEM.CLUSTER.TRANSMIT.MODEL**
- **Well known queue names.**

  SYSTEM.CLUSTER.TRANSMIT.*<CHANNEL-NAME>*



Multiple cluster transmit queues: Manual

- ***Still* configured on the *sending queue manager*, not the owners of the cluster receiver channel definitions.**

- **Administratively define a transmission queue and configure which cluster sender channels will use this transmission queue.**

  DEFINE QLOCAL(GREEN.XMITQ) CLCHNAME(GREEN.*) USAGE(XMITQ)

  - Set a channel name *pattern* in CLCHNAME
  - Single/multiple channels (wildcard)
    - E.g. all channels for a specific cluster (assuming a suitable channel naming convention!)

- **Any cluster sender channel not covered by a manual transmission queue defaults to the DEFCLXQ behaviour**