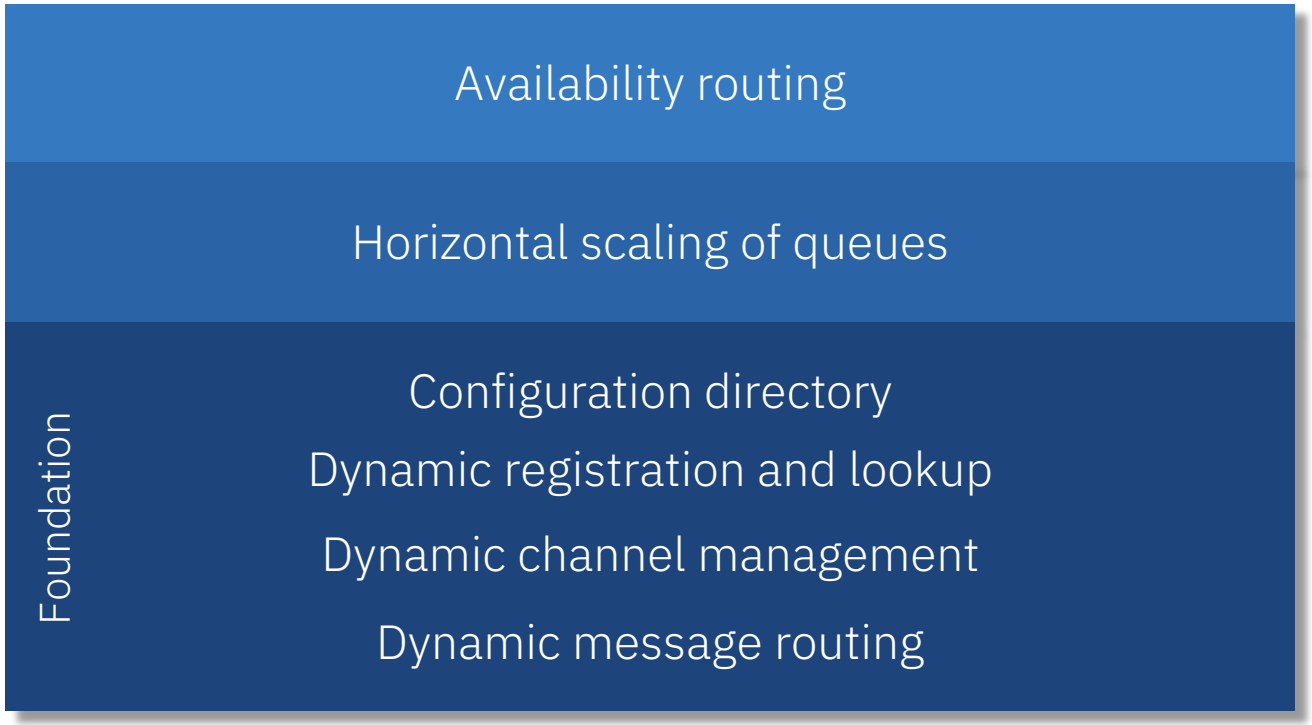
Build Scalability and Availability into Your Messaging Solutions with IBM MQ Clustering

David Ware
Chief Architect, IBM MQ

Celebrating
25
years

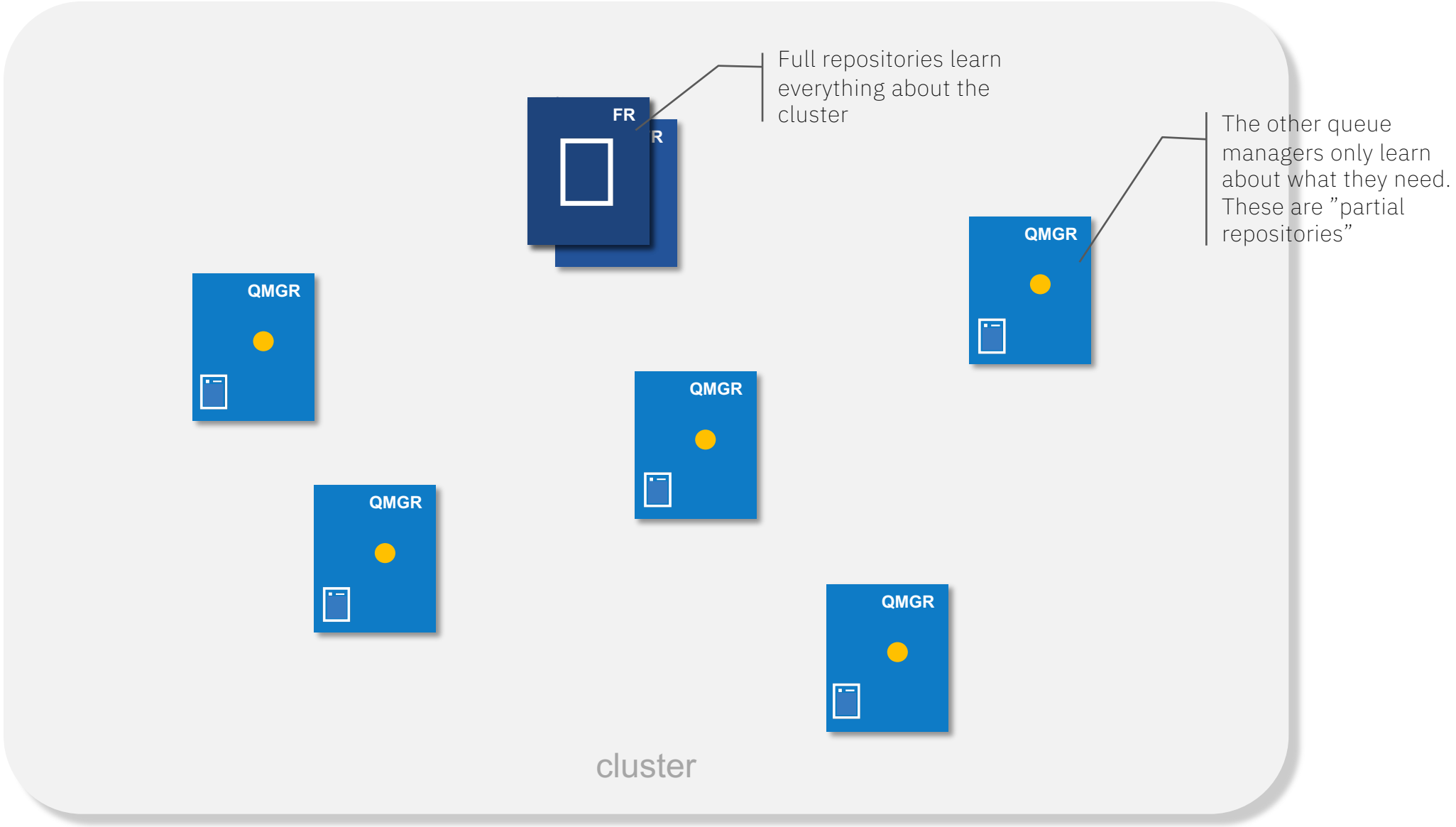
MQ clustering

What MQ Clusters provide :

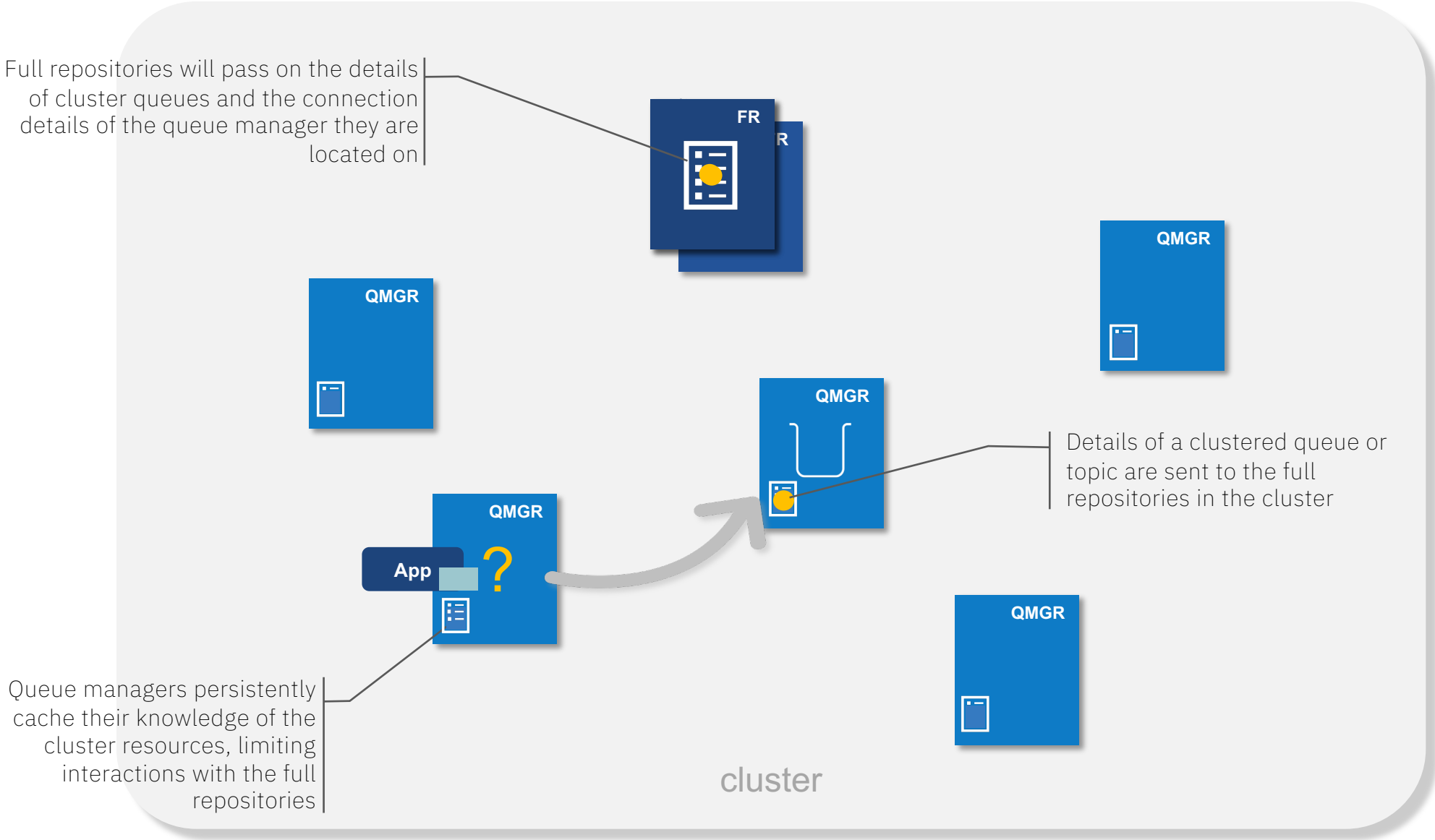


The configuration directory and dynamic routing

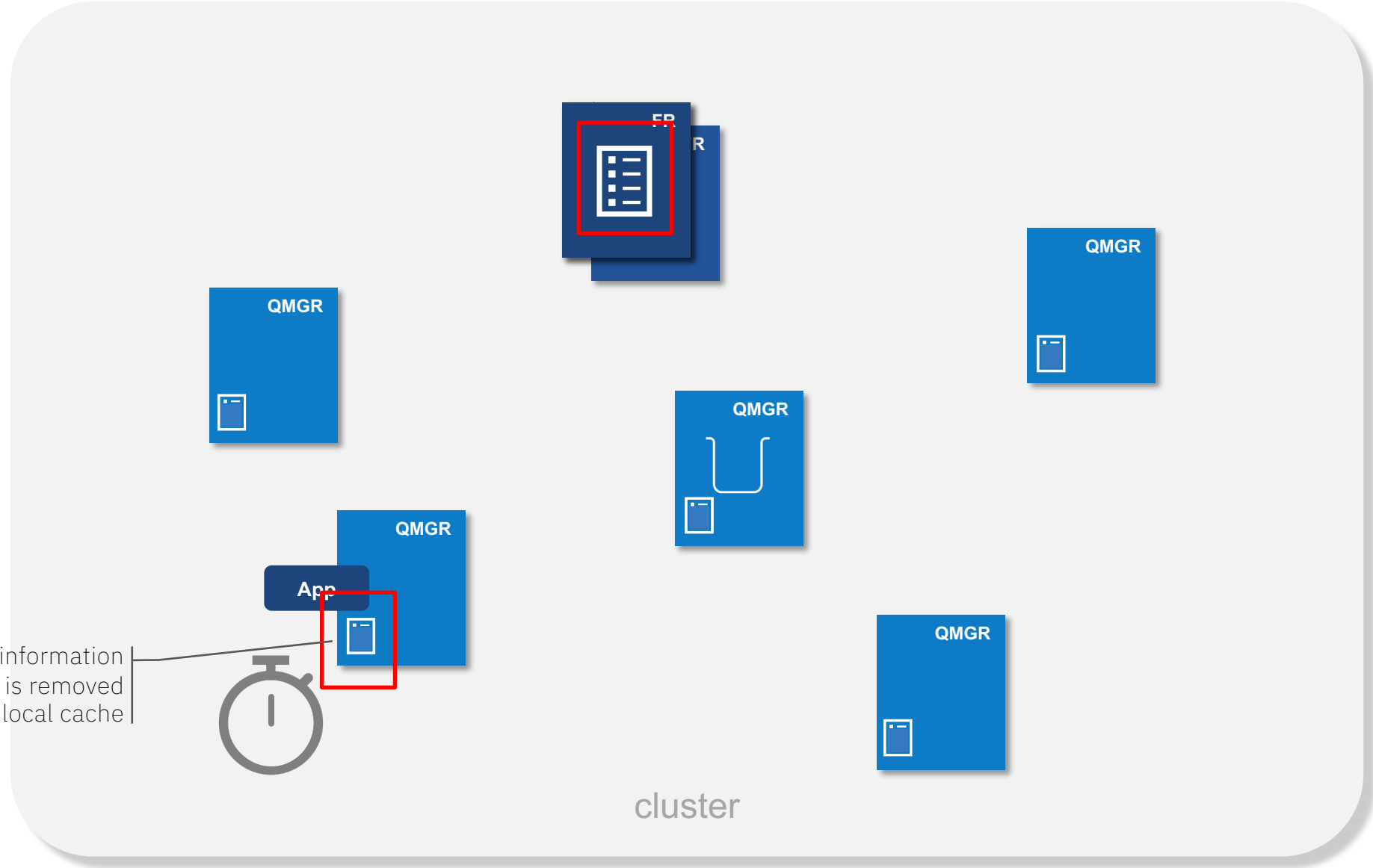
Cluster management



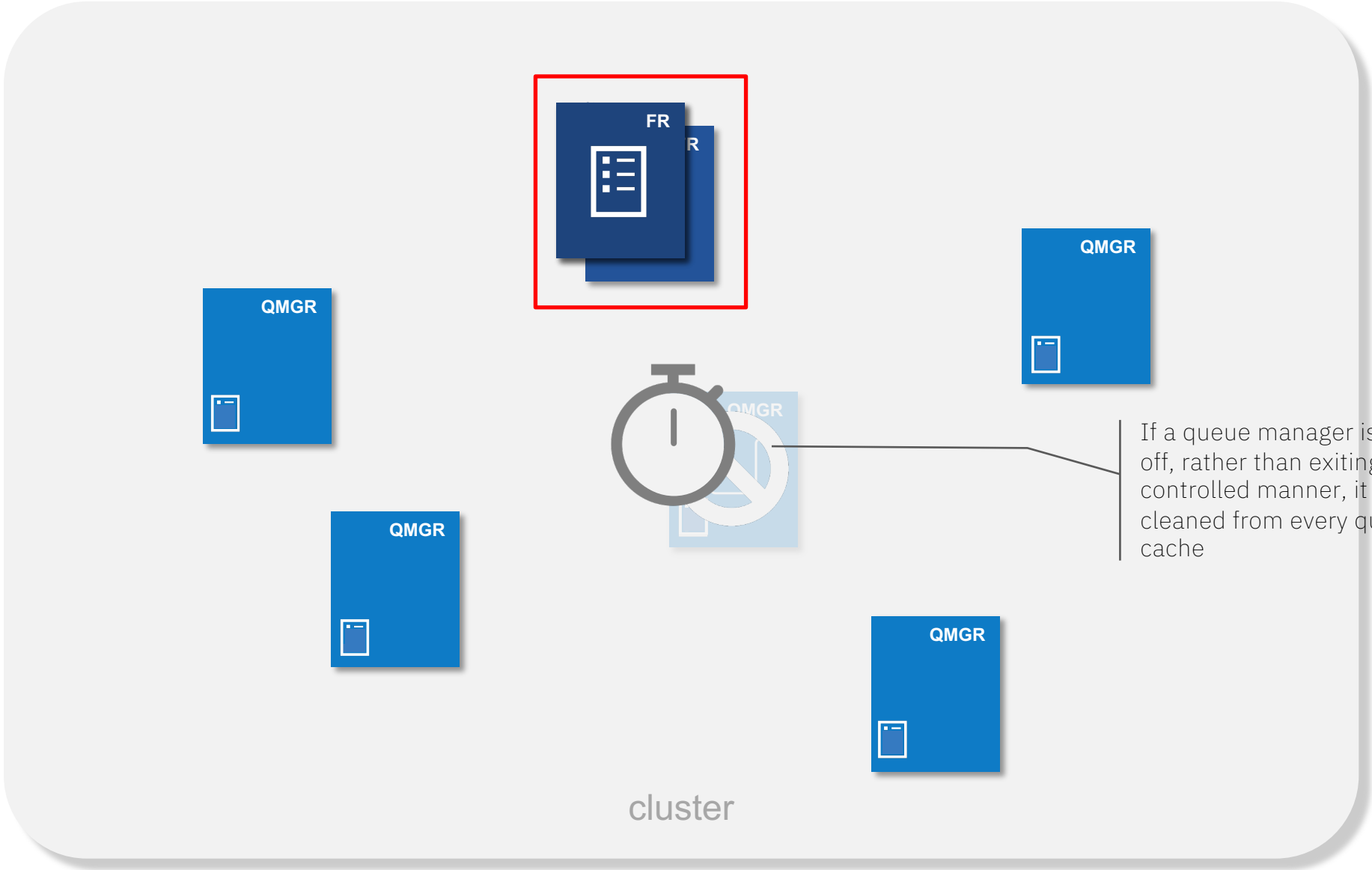
Cluster management



Cluster management



Cluster management

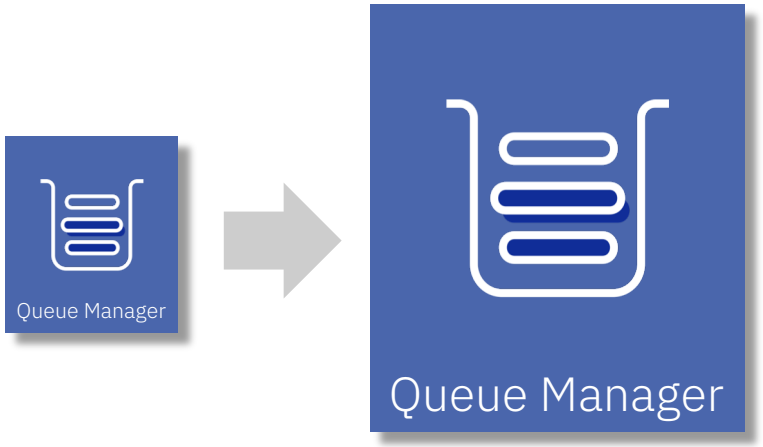


If a queue manager is simply switched off, rather than exiting the cluster in a controlled manner, it will eventually be cleaned from every queue manager's cache

Horizontal scaling

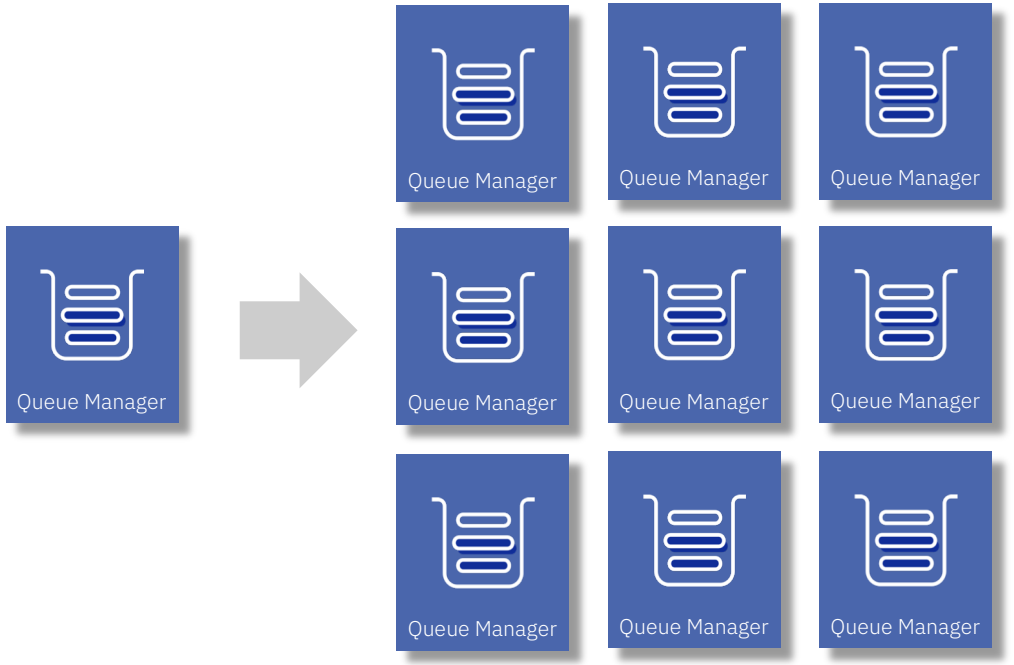
Scaling MQ

Vertical



- Invisible to applications
- Limited by maximum system size
- Liable to hit internal limits
- Not all aspects scale linearly
- Restart times can grow

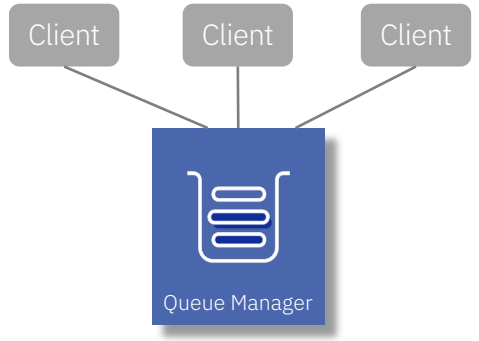
Horizontal



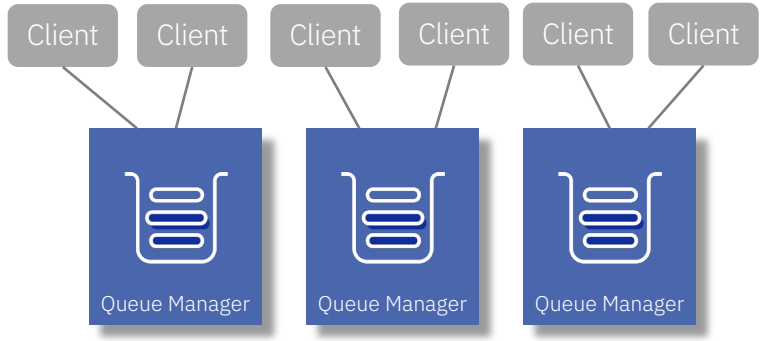
- Unlimited by system size
- All aspects scale linearly
- More suited to cloud scaling
- Reduced restart times
- Enables rolling upgrades
- Visible to applications – limitations apply
- Potentially more complicated

Horizontal scaling MQ

Step 1
Horizontally scale the application into multiple instances, all performing the same role
A queue manager works better when there are multiple applications working in parallel

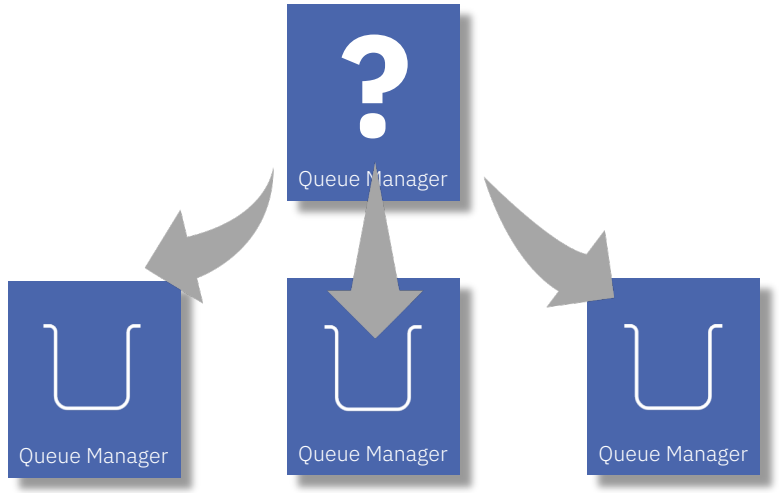


Step 2
Horizontally scale the queue managers
Create multiple queue managers with the ‘same’ configuration
Distribute the application instances across the queue managers



Horizontal scaling with MQ Clustering

- A queue manager will typically route messages based on the name of the target queue
- In an MQ Cluster it's possible for multiple queue managers to independently define the same named queue
- Any queue manager that needs to route messages to that queue now has a choice…



Channel workload balancing

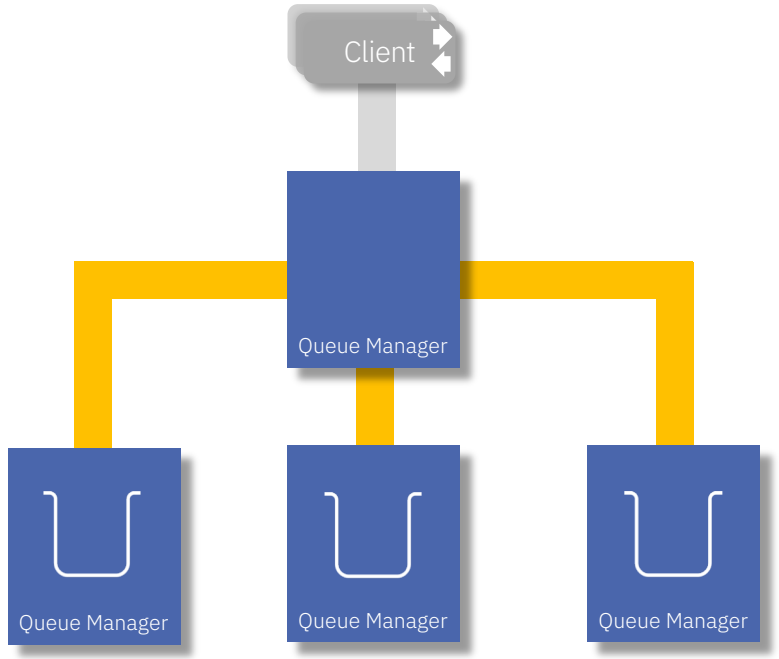
- Cluster workload balancing applies when there are **multiple cluster queues of the same name**

- Cluster workload balancing will be applied in **one of three ways**:
  - When the putting application opens the queue **- bind on open**
  - When a message group is started **- bind on group**
  - When a message is put to the queue **- bind not fixed**

- When workload balancing is applied:
  - The source queue manager builds a list of all potential targets based on the queue name
  - **Eliminates** the impossible options
  - **Prioritises** the remainder
  - If more than one come out equal, **workload balancing** ensues …

- Balancing is based on:
  - The **channel** – not the target queue
  - **Channel traffic** to **all queues** is taken into account
  - **Weightings** can be applied to the channel

- … this is used to send the messages to the chosen target



Channel workload balancing

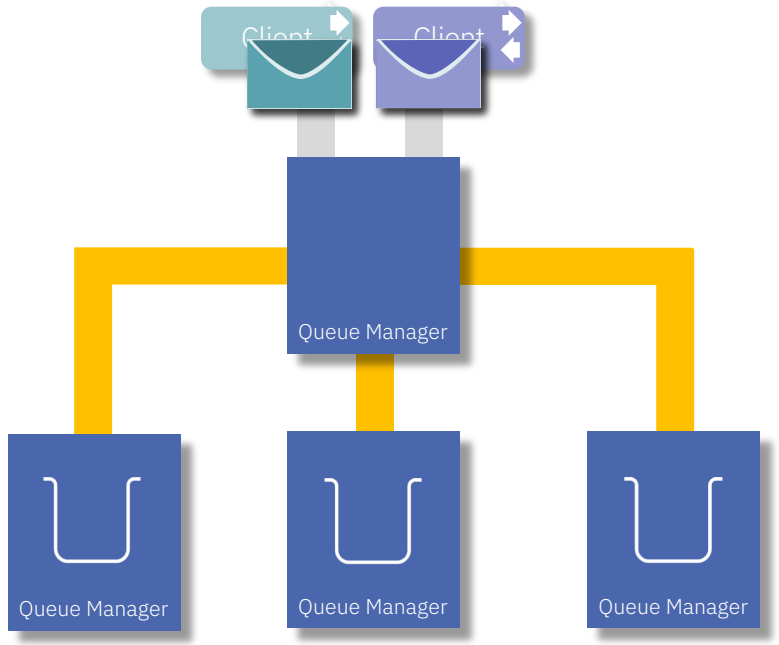
- Cluster workload balancing applies when there are **multiple cluster queues of the same name**

- Cluster workload balancing will be applied in **one of three ways**:
  - When the putting application opens the queue **- bind on open**
  - When a message group is started **- bind on group**
  - When a message is put to the queue **- bind not fixed**

- When workload balancing is applied:
  - The source queue manager builds a list of all potential targets based on the queue name
  - **Eliminates** the impossible options
  - **Prioritises** the remainder
  - If more than one come out equal, **workload balancing** ensues …

- Balancing is based on:
  - The **channel** – not the target queue
  - **Channel traffic** to **all queues** is taken into account
  - **Weightings** can be applied to the channel

- … this is used to send the messages to the chosen target



Channel workload balancing

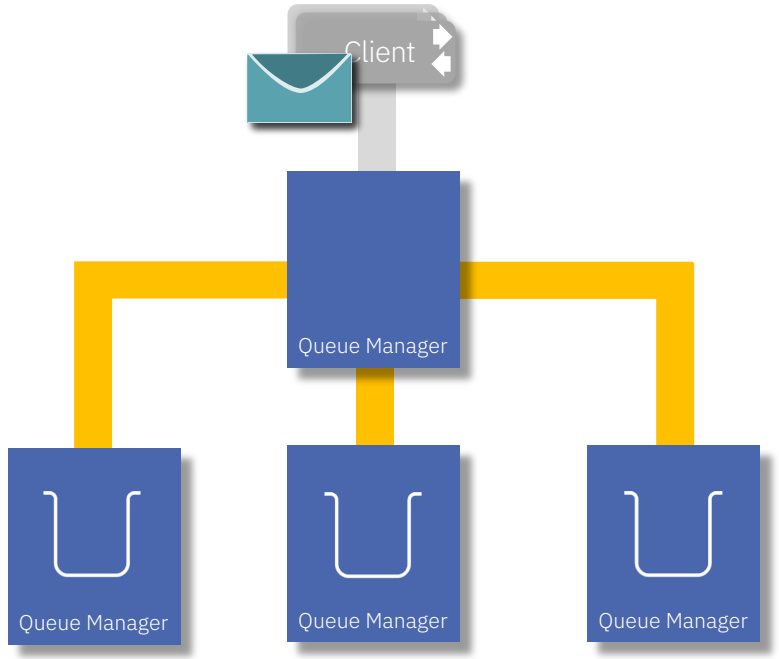
- Cluster workload balancing applies when there are **multiple cluster queues of the same name**

- Cluster workload balancing will be applied in **one of three ways**:
  - When the putting application opens the queue **- bind on open**
  - When a message group is started **- bind on group**
  - When a message is put to the queue **- bind not fixed**

- When workload balancing is applied:
  - The source queue manager builds a list of all potential targets based on the queue name
  - **Eliminates** the impossible options
  - **Prioritises** the remainder
  - If more than one come out equal, **workload balancing** ensues …

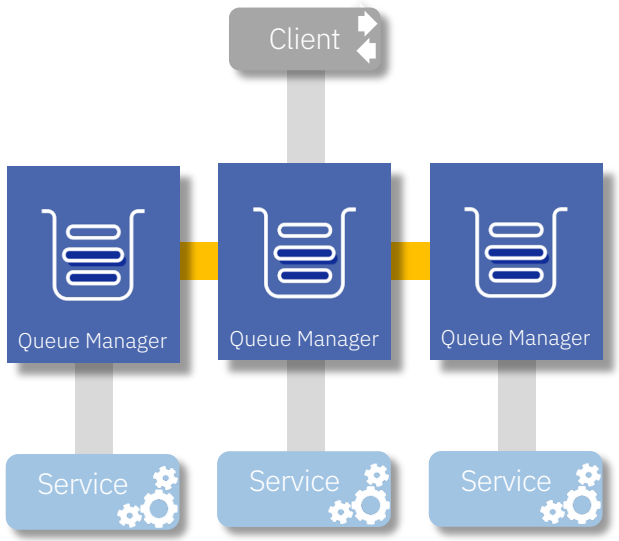
- Balancing is based on:
  - The **channel** – not the target queue
  - **Channel traffic** to **all queues** is taken into account
  - **Weightings** can be applied to the channel

- … this is used to send the messages to the chosen target



Horizontal scaling – *do I really need MQ Clustering?*

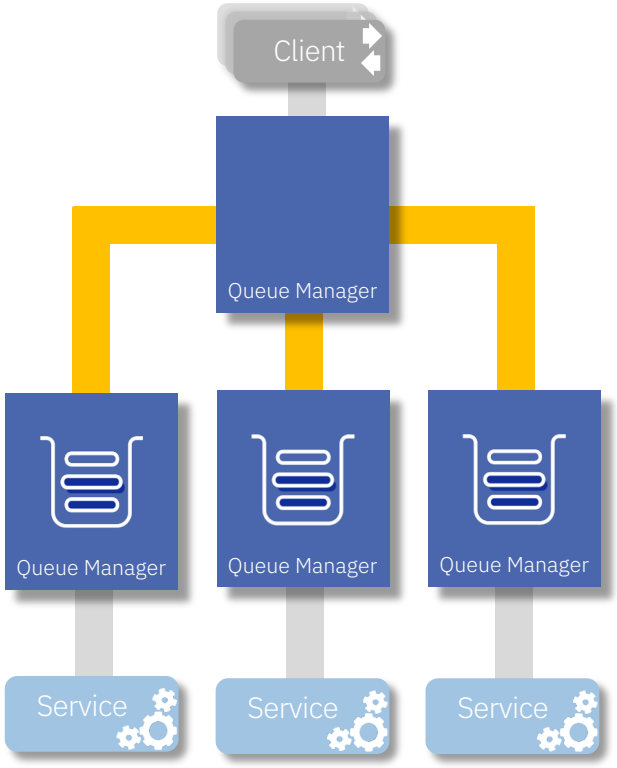
Single producing application



Q. Is clustering required?

A. Definitely

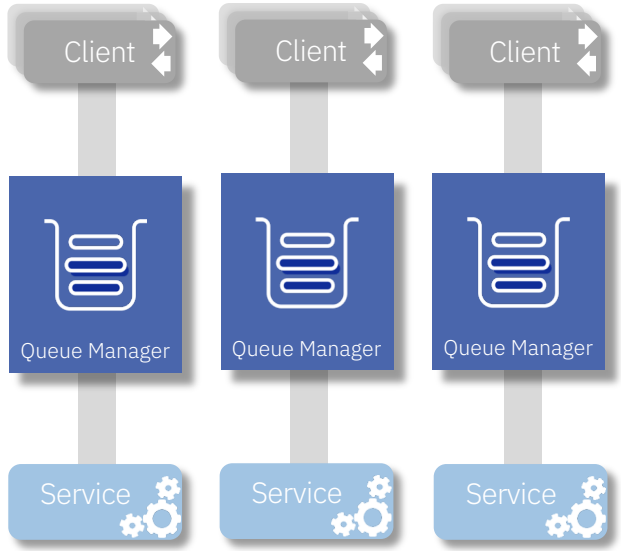
Gateway routing



Q. Is clustering required?

A. Definitely

Scaled out applications



Q. Is clustering required?

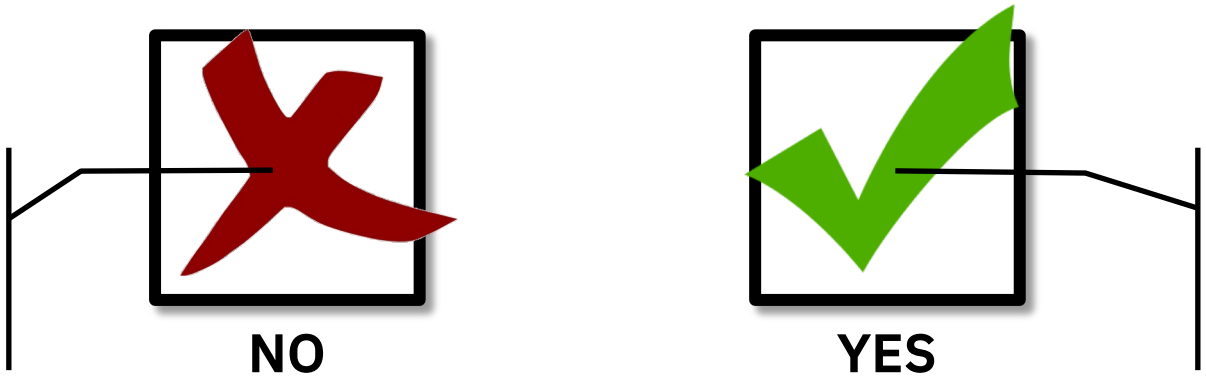
A. Maybe, maybe not ...

Availability routing

Clustering for availability

Is MQ Clustering a high availability solution?

Not for the message data. Each message is only available from a single queue manager



Clustering can form a part of the overall high availability of the messaging system

- Having multiple potential targets for any message can improve the availability of the solution, always providing an option to process new messages.
- A queue manager in a cluster has the ability to route new and old messages based on the **availability of the channels**, routing messages to running queue managers.
- Clustering can be used to route messages to active consuming applications.

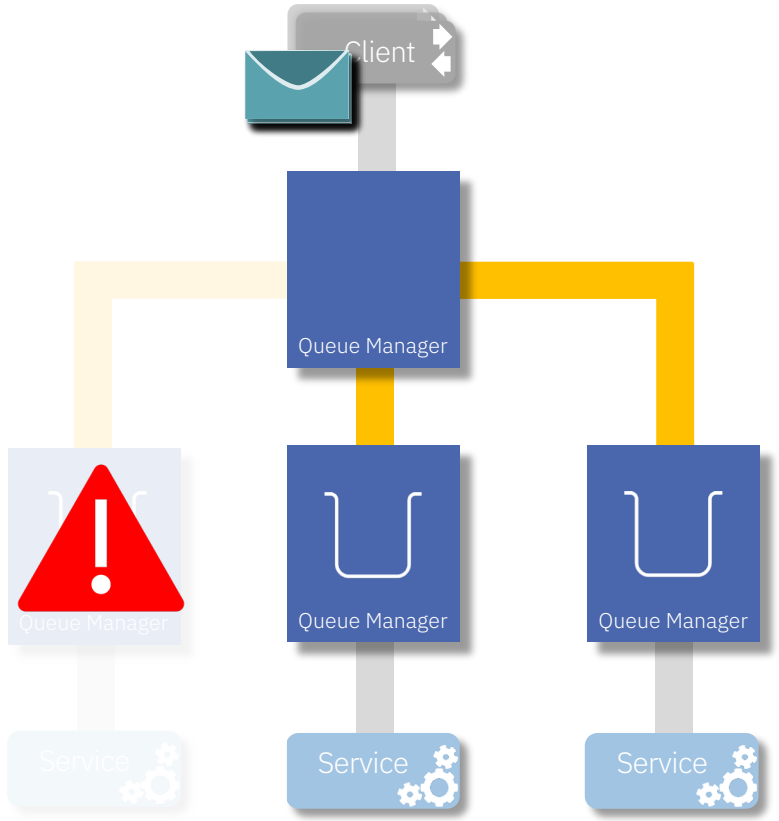
Channel availability routing

- When performing workload balancing, the availability of the channel to reach the target is a factor
- All things being equal, messages will be routed to those targets with a working channel

Routing of messages based on availability doesn't just happen when they're first put, it also occurs for **queued transmission messages** every time the channel is **retried** So blocked messages can be re-routed, if they're not prevented...

Things that can prevent routing

- Applications targeting messages at a specific queue manage (e.g. reply message)
- Using "cluster workload rank"
- Binding messages to a target



Pros and cons of binding

Bind on open / Bind on group

Bind context:
Duration of an *open*
Duration of *logical group*

- All messages put within the bind context will go to same target*
- Message order can be preserved**
- Workload balancing logic is only driven at the start of the context

- Once a target has been chosen it **cannot change**
  - Whether it's available or not
  - Even if all the messages could be redirected

Bind not fixed

Bind context:
None

- Greater *availability,* a message will be redirected to an available target***

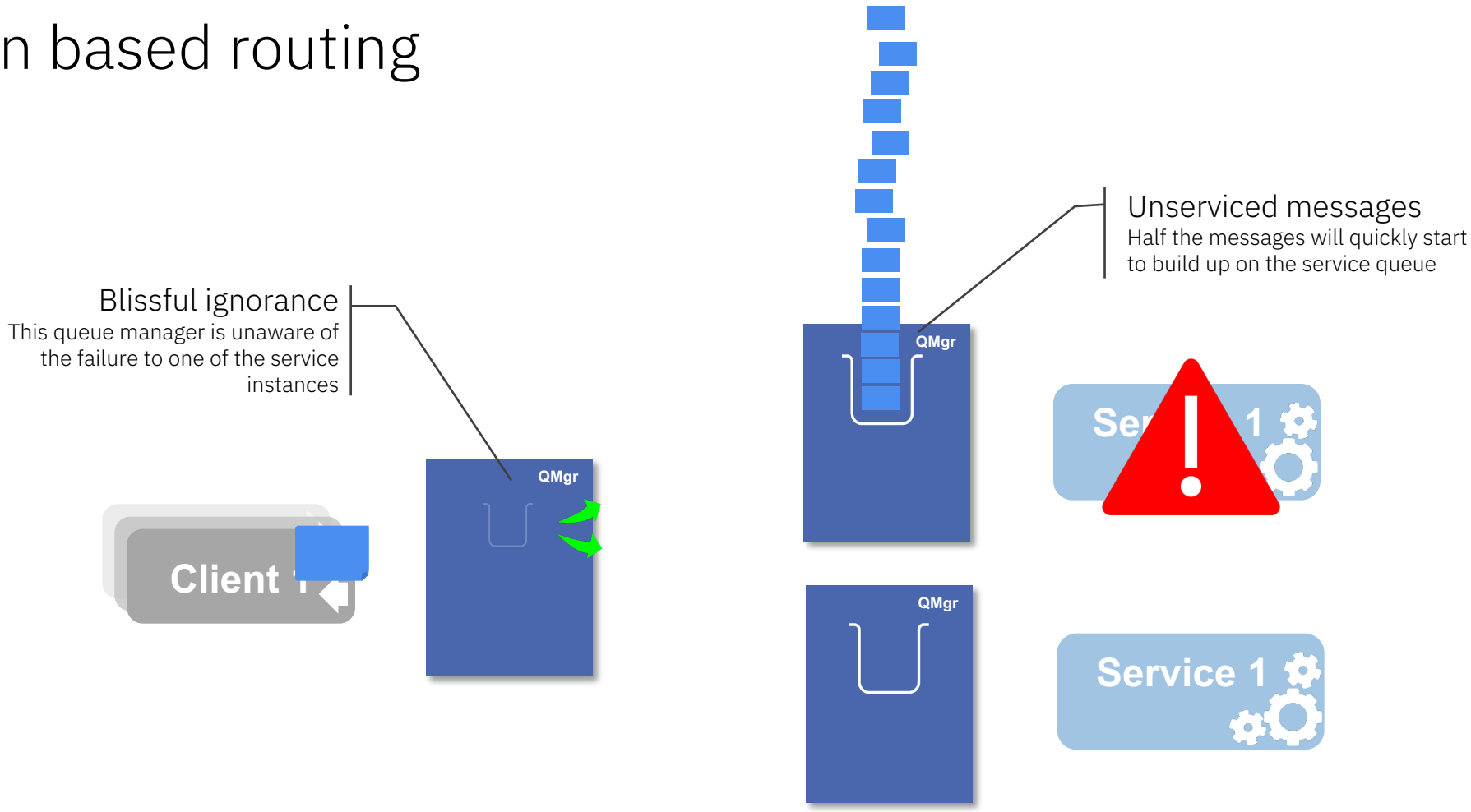
- Overhead of workload balancing logic for every message
- Message order may be affected

Bind on open is the default
It could be set on the cluster queue (don't forget aliases) or in the app

* While a route is known by the source queue manager, it won't be rebalanced, but it could be DLQd
** Other aspects may affect ordering (e.g. deadletter queueing)
*** Unless it's fixed for another reason (e.g. specifying a target queue manager)

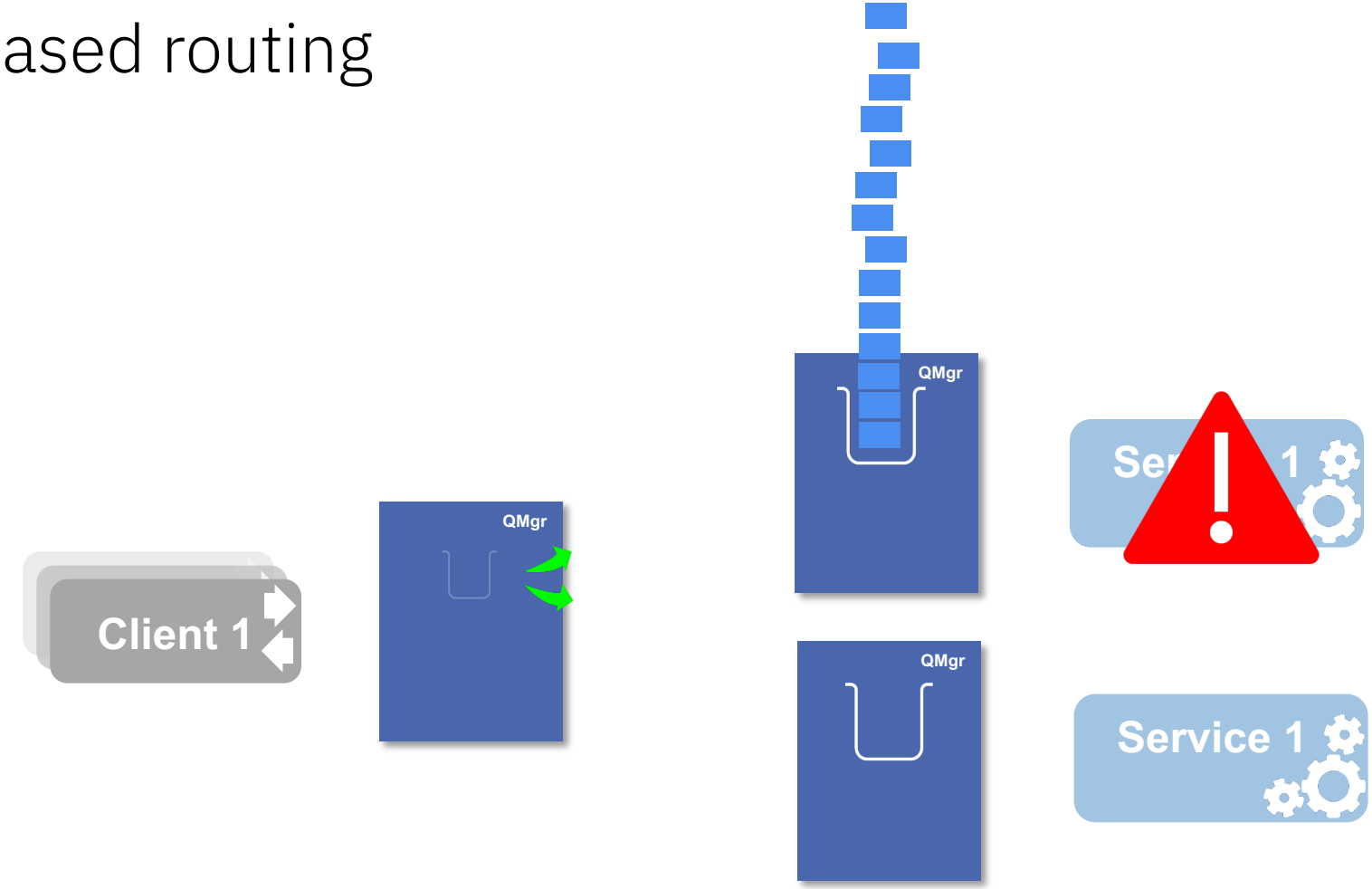
Application availability routing

Application based routing

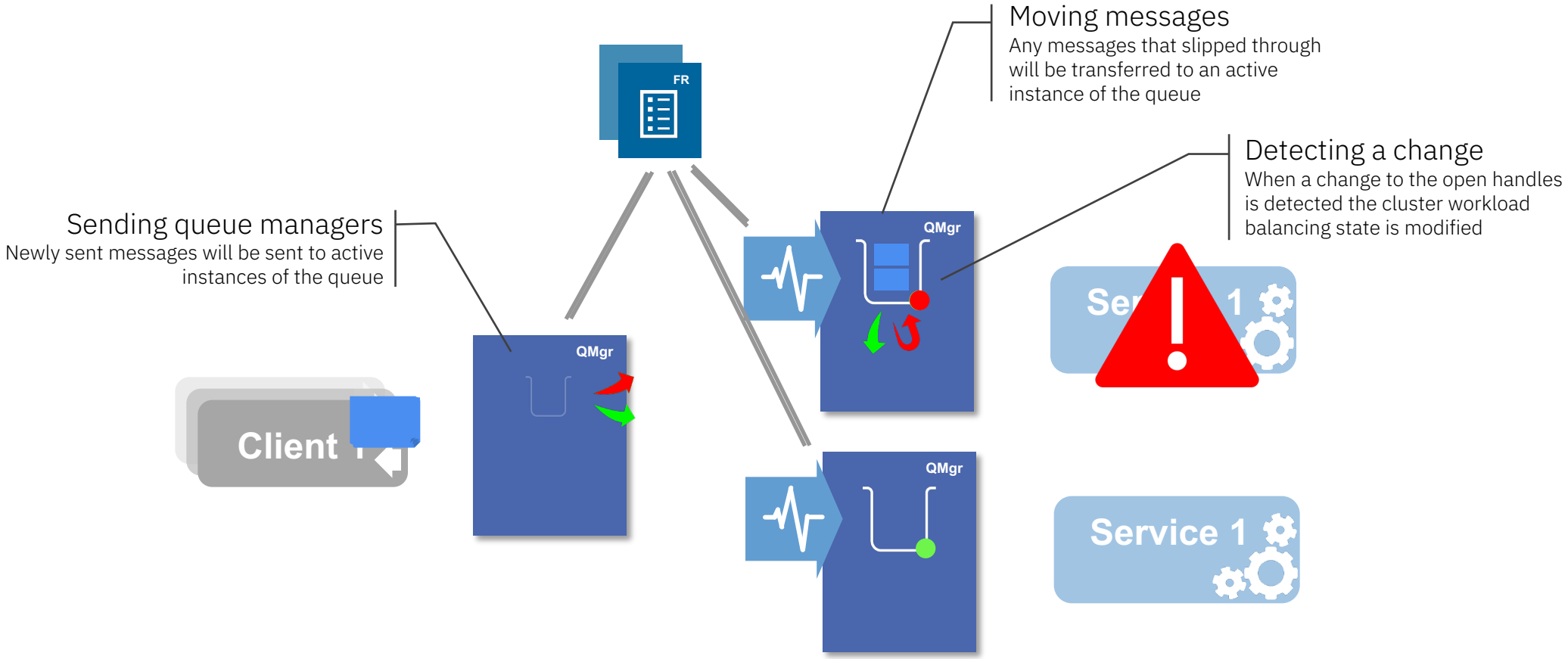


- Cluster workload balancing does not take into account the availability of receiving applications
- Or a build up of messages on a queue

Application based routing



Application based routing

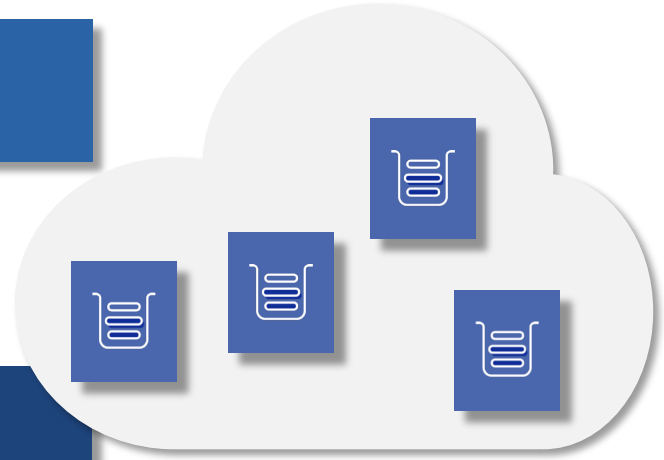


- MQ provides a sample monitoring service tool, **amqsclm**
- It regularly checks for attached consuming applications (IPPROCS)
- And automatically adjusts the cluster queue definitions to route messages intelligently (CLWLPRTY)
- That information is automatically distributed around the cluster

What about in *the cloud*?

MQ Clusters and Clouds

“Cloud platforms provide all an MQ cluster can do”



Not quite…

Clouds often provide cluster-like capability:
Directory services and routing
Network workload balancing

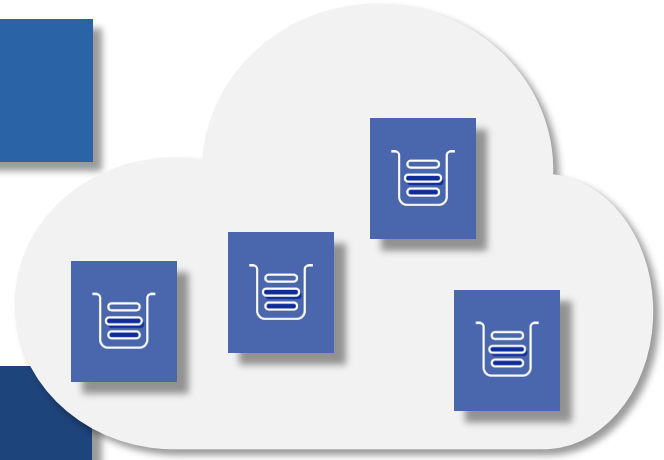
Great for stateless workload balancing
Can be good for balancing unrestricted clients across multiple

But where state is involved, such as reliably sending messages from one queue manager to another without risking message loss or duplication, such routing isn’t enough

That’s still the job of an MQ cluster…

MQ Clusters and Clouds

“So will MQ clusters work in a cloud?”



Yes, but think it through first…

Some things may be different in your cloud:

- A new expectation that queue managers will be created and deleted more dynamically than before
- Queue managers will “move around” with the cloud
- Your level of control may be relaxed

Adding and removing queue managers

Adding queue managers

- Have a simple clustering topology
- Separate out your full repositories and manage those separately
- Automate the joining of a new queue manager

Removing queue managers - *this is harder!*

- You might have messages on it that you need, how are you going to remove those first?
- If you just switch off the queue manager, it'll still be known by the cluster for months!
- Is that a problem?
  - If routing is based on availability, messages will be routed to alternative queue managers
  - Messages will sit on the cluster transmission queues while the queue manager is still known
  - It makes your view of the cluster messy
- Automate the cluster removal
  - From the deleting queue manager
    - Stop and delete the cluster channels – give it a little time to work through to the FRs
  - Then from a full repository – as it'll never be coming back
    - RESET the queue manager out of the cluster

Queue managers will move around

- Usually this is not actually a problem, clouds are doing a good job of hiding that from you
  - e.g. Kubernetes “services” are effectively providing DNS for a container

- Check the cloud capabilities available to you
- Probably best to use hostnames in the connection details rather than IP addresses
- Remember that often names/addresses may only be visible from within the cloud
  - External addresses may differ, comma separated CONNAMEs can help here

Control over the cluster

Keep it simple

In a potentially self service world, where new applications need their own queue manager, overall control of configuring everything in the cluster is not desirable. To prevent chaos don't just pick up your bespoke, on-prem, cluster topology and put it in the cloud – re-think it:

- Simplify the cluster topology, do you really need all those overlapping clusters?
- Make sure the cluster infrastructure (the full repositories, gateway queue managers) are still under your control
- Script the joining/leaving a cluster and automate this with the deployment of new queue managers
- A cluster is a shared namespace, enforce naming conventions on cluster queues and topics
- Use your full repositories as a view across the cluster

Clustering good practices

Full repositories

- Dedicate a pair of queue managers to being the full repositories for the cluster
  - Don’t even think of them as queue managers
- A pair for redundancy obviously, but why not more?
  - More than two won’t work how you hope it will…
    - A queue manager will randomly pick two full repositories to work with (manually defining your sender channels doesn’t guarantee which two)
    - If those two are unavailable it won’t go off and find another one

Check your defaults

Cluster transmission queues **DEFCLXQ**

- MQ started with each queue manager having just the one
- It added the ability to automatically split that out into one per cluster channel
  - This improves visibility of channel problems
  - And reduces the cross-channel impacts of a blocked channel

Cluster workload rank and priority **CLWLRANK/CLWLPRTY**

- These are used as a tie breaker across multiple queues/channels to determine where messages are sent
- They default to zero (the lowest), so you can raise one to make it more favourable
- But you can't lower one to make it less favourable
- Consider setting them all to be around the middle (e.g 5) to give yourself options

Default application bind setting **DEFBIND**

- The default is "on open"
- Great if you need affinity across messages, poor if you value message availability over that
- Consider setting this to "not fixed"

Problem diagnosis

Problem diagnosis

Cluster queues are not visible where you think they should be, or messages sit on a transmission queue for no reason...

It's often:

- A break in communication between queue managers
  - Check the PR->FR and FR->PR channels
- A config issue
  - Check the host of the resource and that the queue manager has correctly joined the cluster

What to check:

- Configuration – double check the cluster names
- Queue manager cluster knowledge – are the queue managers known to the FRs?
- Queue manager channels – check the PR->FR and FR->PR channels
- Queue manager error logs – look for messages about expiring objects
- System queues - ...

Sequence of diagnosis steps:

- Check the queue manager where the cluster definitions live
- Check all the full repositories
- Check the queue manager where you're sending the messages from

Know your system cluster queues

SYSTEM.CLUSTER.COMMAND.QUEUE

- Source of all work for the cluster repository process
- Messages arrive here from local configuration commands and from FR->PR and PR->FR
- If you delete messages from here you may need a cluster refresh
- Steady state: **empty**

SYSTEM.CLUSTER.REPOSITORY.QUEUE

- Persistent store for accumulated cluster knowledge
- All cluster object changes persisted here
- Based on local configuration and remote knowledge
- If you delete messages from here your only option is a cluster refresh
- Steady state: **non-empty**

SYSTEM.CLUSTER.TRANSMIT.*

- Used to transfer all messages within a cluster
- Shared between user messages and cluster control messages
- If you delete messages from here you may need a cluster refresh, and you may have lost your own messages!
- Steady state: **Ideally empty** (may contain messages whilst channels are down)

SYSTEM.CLUSTER.HISTORY.QUEUE

- Intended for capturing diagnostics for IBM MQ Service teams
- Only used if REFRESH CLUSTER is issued
- Entire contents of local cluster cache written prior to processing REFRESH
- Messages expire after 180 days
- Steady state: **ignore it**

Celebrating

25 years

Clustering quiz

Quiz

“All the queue managers should be at the same level”

False – MQ does an excellent job at interoperating between versions
Recommendation – stay in support, we only test the in-support combination of queue managers

Quiz

“Full repositories must be the highest version in the cluster”

False(ish) – this is a recommendation, not a rule…
Recommendation – when new features are added in MQ versions, for them to work across queue managers in a cluster, typically you’ll need the sending and receiving **and full repository** queue manages to be at that later level. If you leave your full repositories behind, the feature may not work. And if you upgrade it later, it may take weeks for the capability to filter through (dependent on the cluster expiry cycle)

Quiz

“There can only be two full repositories at any time”

False(ish) – this is a recommendation, not a rule…
Recommendation – steady state should be two full repositories, but if you’re migrating, don’t worry, dropping to one or growing to three is perfectly acceptable

Quiz

“If I need message ordering I can’t use clustering”

False – Many factors can affect message ordering, cluster is just another one
Recommendation – look at all the points that messages may be re-ordered (e.g. applications, DLQs, transactions,…) and make sure you consider how the cluster works. For example, do you allow messages to be workload balanced? Are messages bound to prevent reallocation?

Quiz

“When I make a clustering change, REFRESH CLUSTER is needed to propagate it”

FALSE! – REFRESH CLUSTER will cause a break in your cluster knowledge, affecting message routing, don’t just do it for the sake of it. The cluster takes care of updates, if a change is made to the cluster configuration (e.g. a new channel or queue is created), this will be propagated to the full repositories immediately, leave it to the cluster.
Recommendation – If that’s not what you’re seeing, raise a PMR!

Quiz

“NEVER use REFRESH CLUSTER”

False – there’s always an exception to the rule…
Recommendation – sometimes the cluster will need a little help. For example, if you restore a queue manager from a backup (or move to an asynchronously replicated DR copy) it doesn’t know this, and it may have missed an update from the FRs. Here a refresh can force the sharing of the latest information

Quiz

“It’s best to manually define CLUSSDR channels everywhere”

False – Once the recipient’s CLUSRCVR information is received, and manual definition is ignored
Recommendation – Only have a single CLUSSDR, to one of the full repositories. The rest will be discovered.

Celebrating

25 years