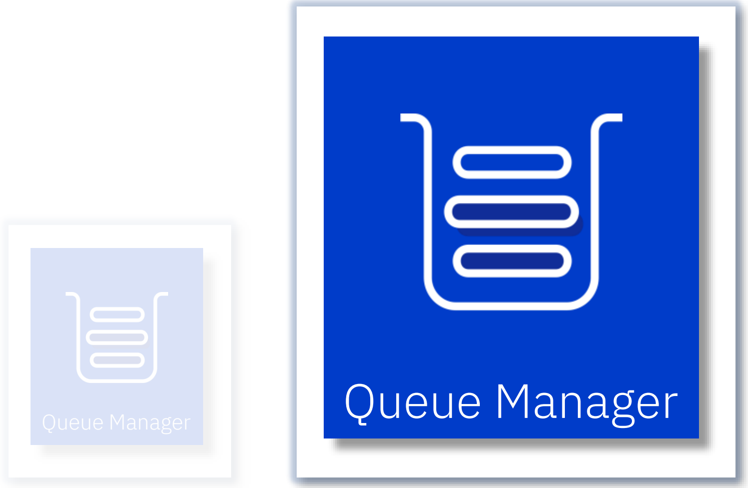
Integration Technical Conference 2019

Build a fault tolerant and scalable IBM MQ solution

David Ware
Chief Architect, IBM MQ

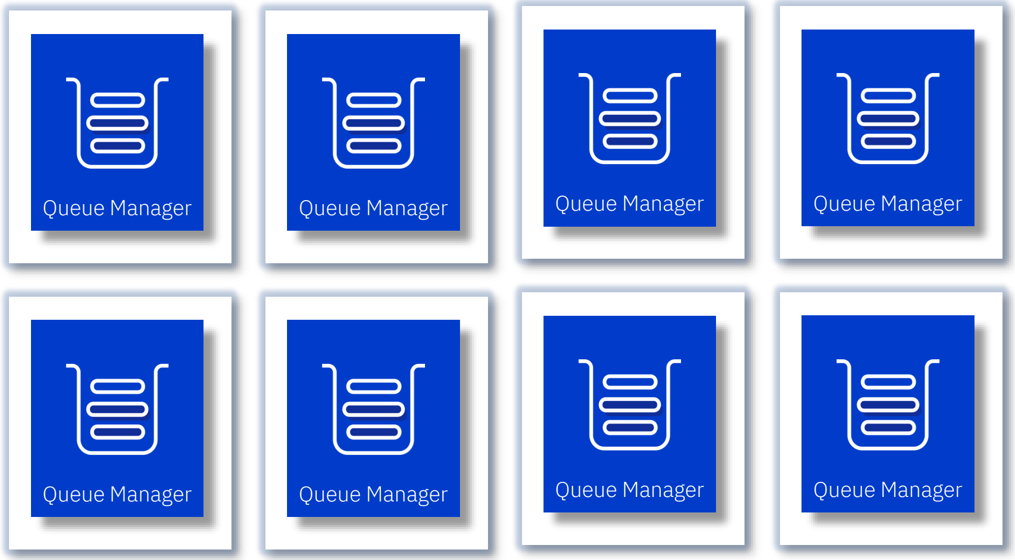
IBM **Cloud**

Scaling



Single

x4



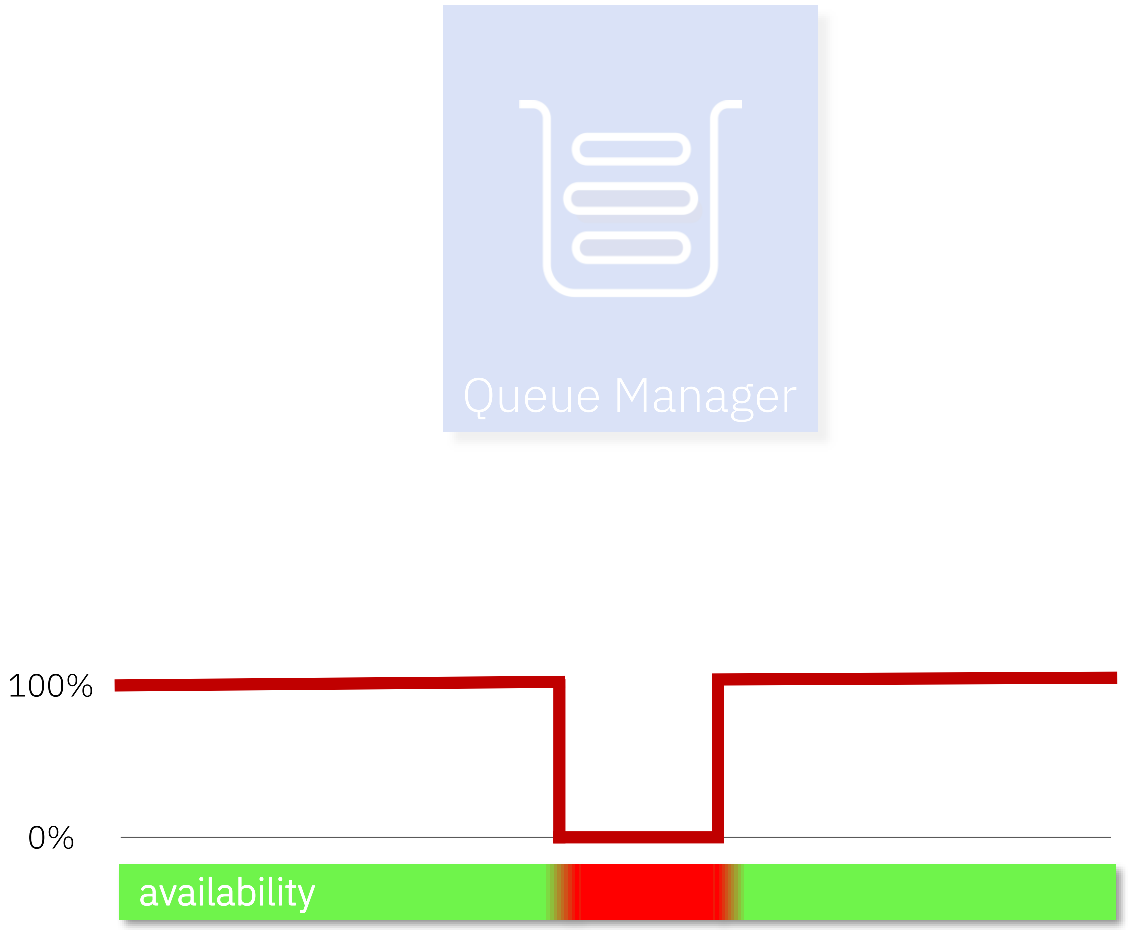
Multiple

xn

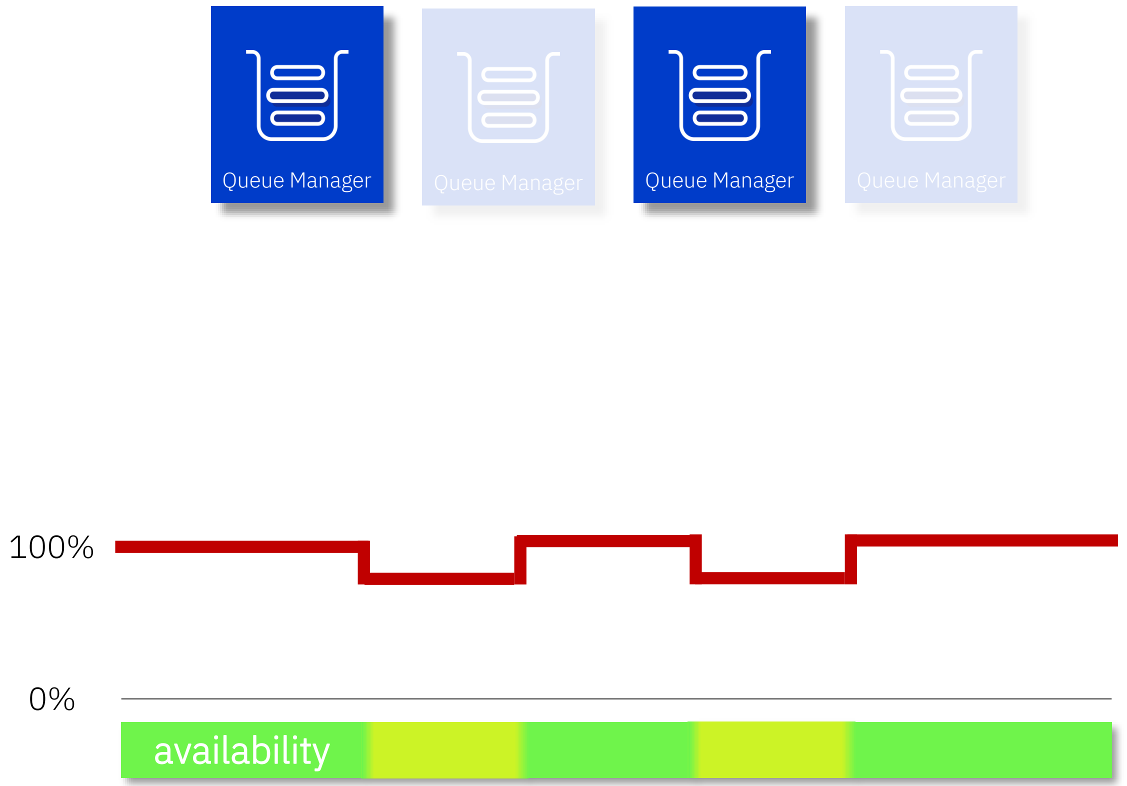
© 2019 IBM Corporation

Fault Toleration

Single



Multiple



© 2019 IBM Corporation

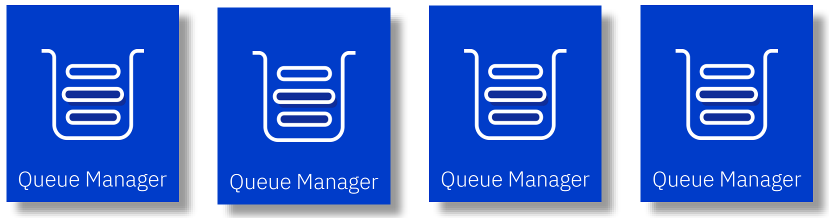
Single vs. Multiple

Single

Queue Manager

- ○ Simple
- ○ Invisible to applications
- ○ Limited by maximum system size
- ○ Liable to hit internal limits
- ○ Not all aspects scale linearly
- ○ Restart times can grow
- ○ Every outage is high impact

Multiple

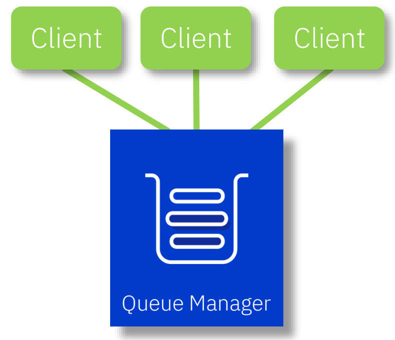


- ○ Unlimited by system size
- ○ All aspects scale linearly
- ○ More suited to cloud scaling
- ○ Reduced restart times
- ○ Enables rolling upgrades
- ○ Tolerate partial failures
- ○ Visible to applications – limitations apply
- ○ Potentially more complicated

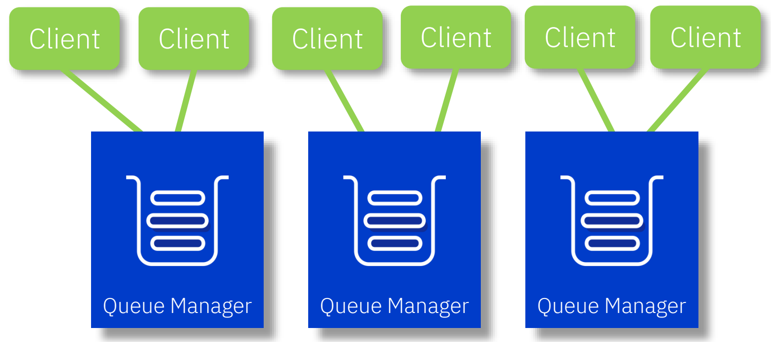
© 2019 IBM Corporation

It’s not just the queue managers…

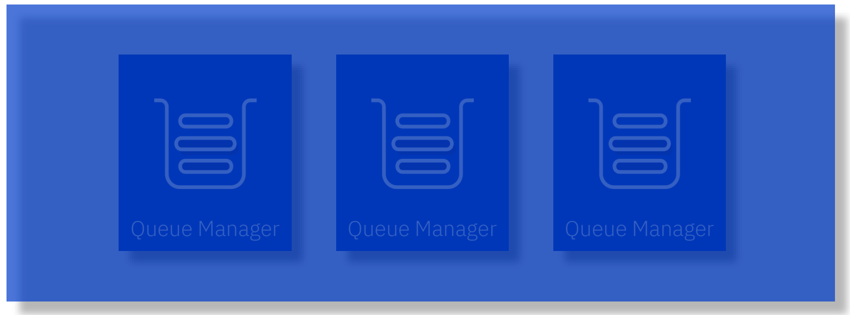
Step 1
Horizontally scale the application into multiple instances, all performing the same role
A queue manager works better when there are multiple applications working in parallel



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



© 2019 IBM Corporation



Try to stop thinking about each individual queue manager and start thinking about them as a ~~cluster~~

~~group~~

~~collective~~

uniform cluster

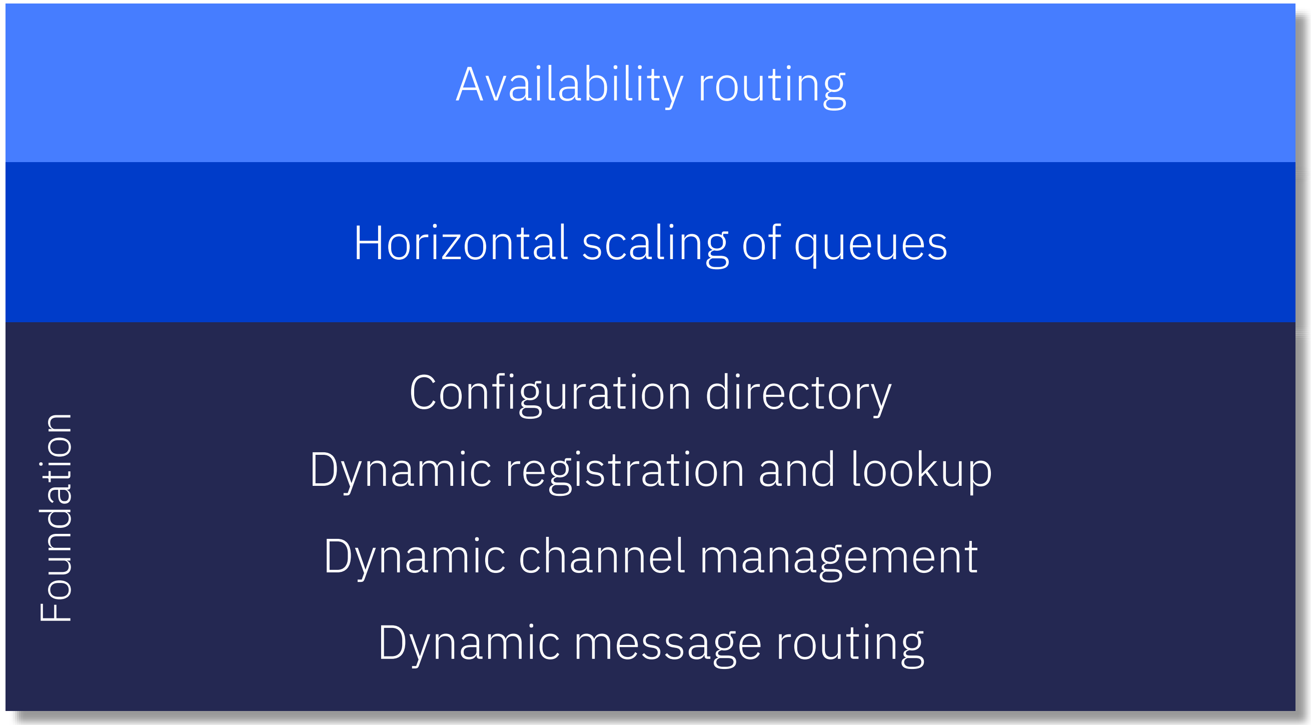
maybe....

© 2019 IBM Corporation

MQ Clustering

MQ clustering

What MQ Clusters provide :


Availability routing
Horizontal scaling of queues
Foundation
Configuration directory
Dynamic registration and lookup
Dynamic channel management
Dynamic message routing

© 2019 IBM Corporation

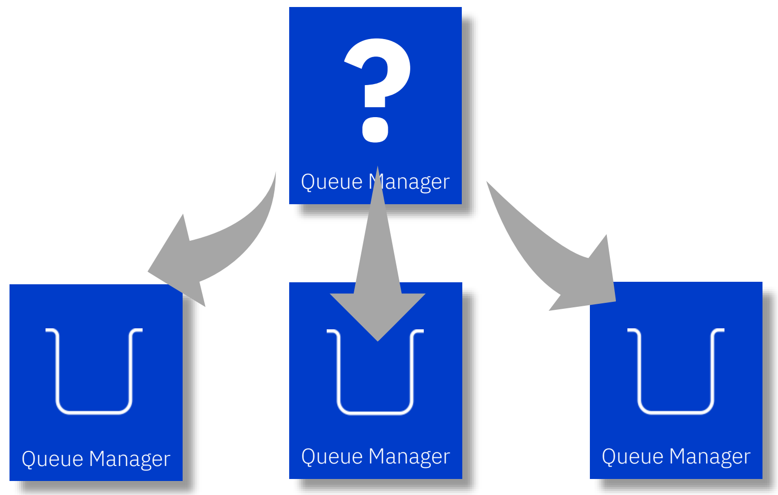
Horizontal scaling with MQ Clustering

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 is 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…



© 2019 IBM Corporation

Channel workload balancing

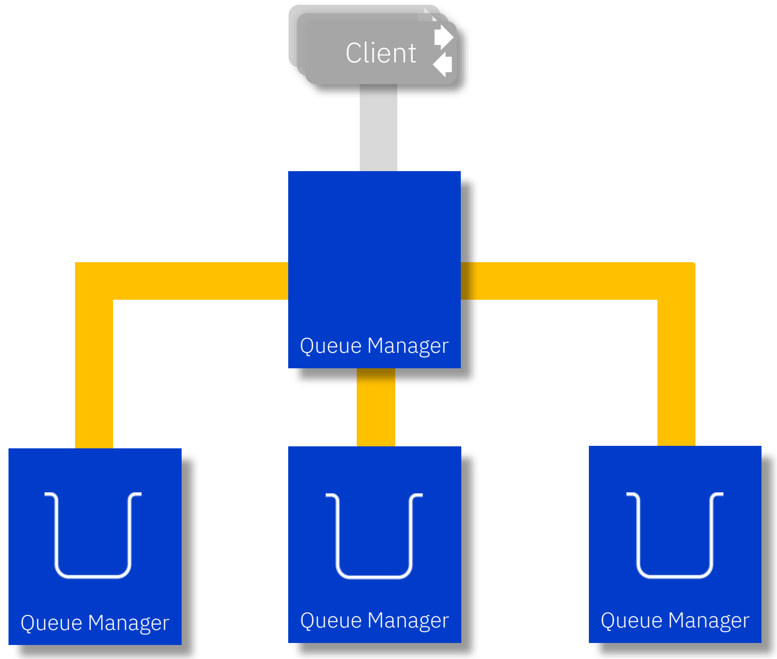
- Cluster workload balancing applies when there are **multiple cluster queues of the same name**

- 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

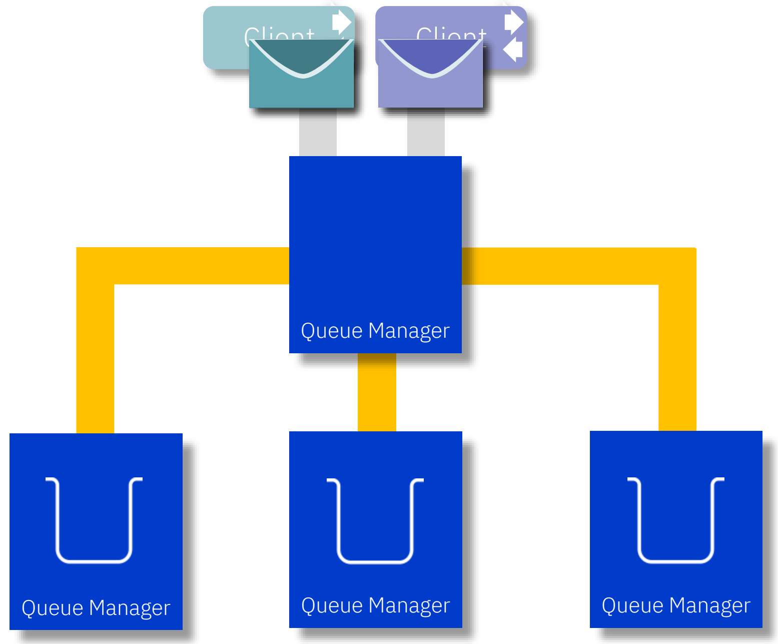
- ... this is used to send the messages to the chosen target



© 2019 IBM Corporation

Channel workload balancing

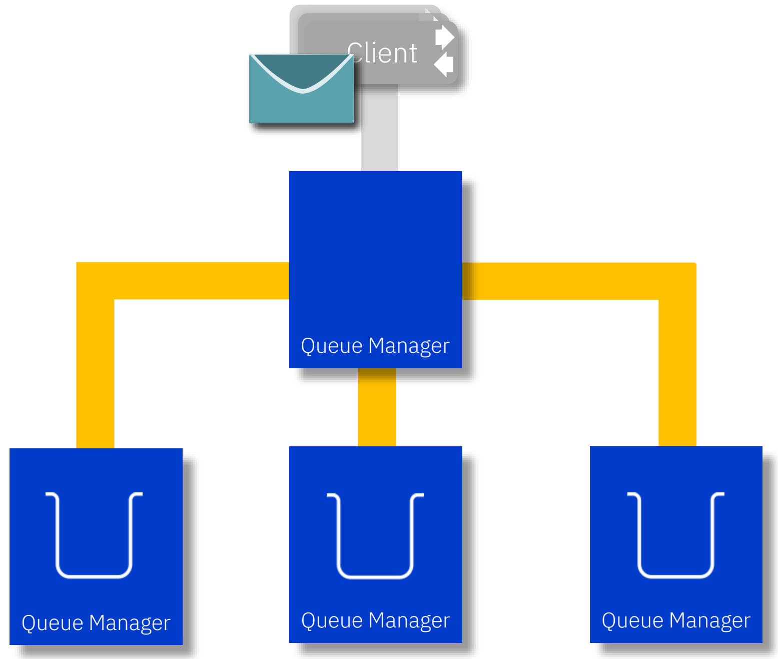
- Cluster workload balancing applies when there are **multiple cluster queues of the same name**
- Cluster workload balancing will be applied in **one of three ways**:
  - When the putting application opens the queue **- bind on open**
  - When a message group is started **- bind on group**
  - When 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



© 2019 IBM Corporation

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



© 2019 IBM Corporation

Channel workload balancing

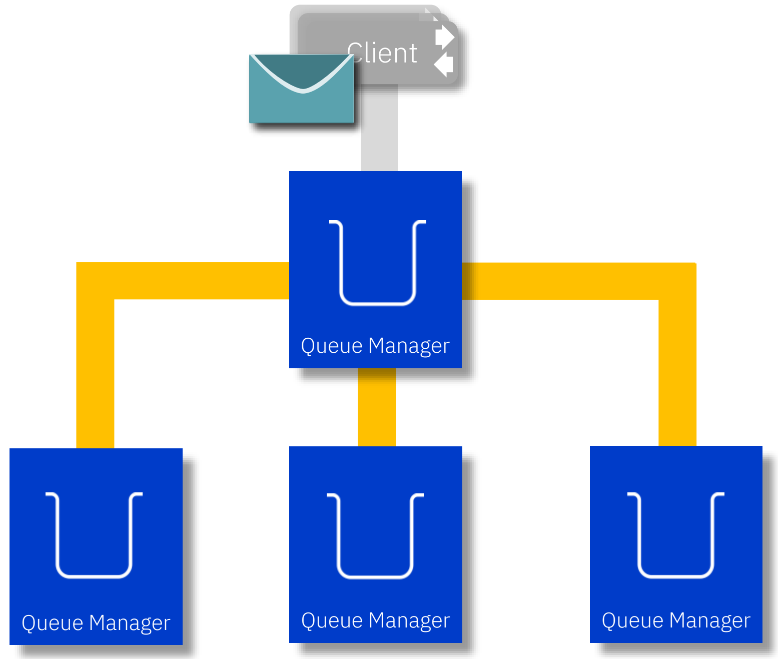
- Cluster workload balancing applies when there are **multiple cluster queues of the same name**

- 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

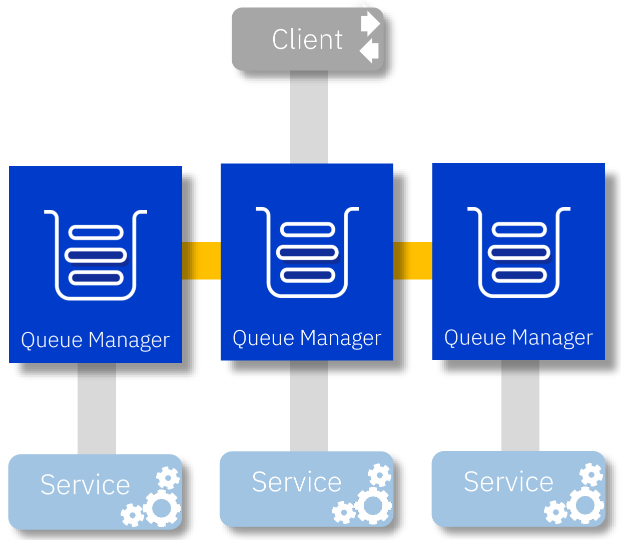


Tip: By default, a matching queue on the same queue manager that the application is connected to will be prioritized over all others for speed. To overcome that, look at *CLWLUSEQ*

© 2019 IBM Corporation

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

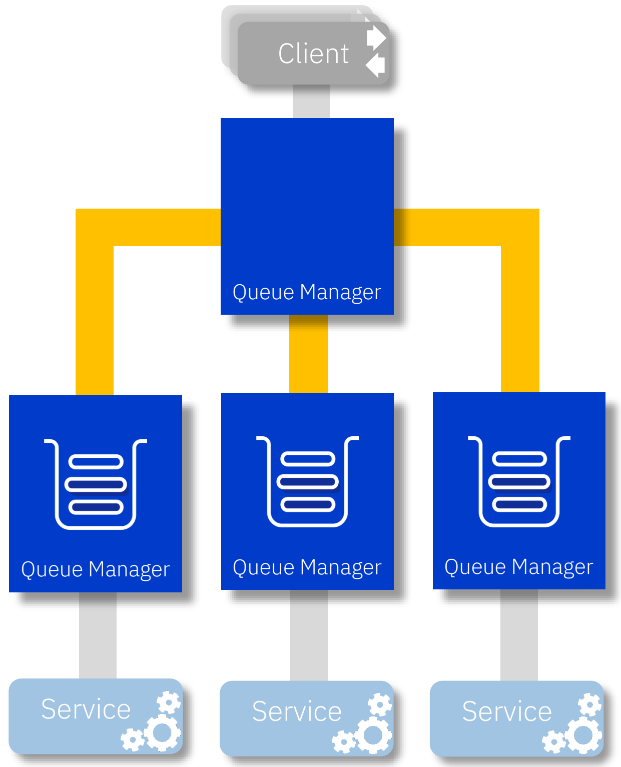
Single producing application



Q. Is clustering required?

A. Definitely

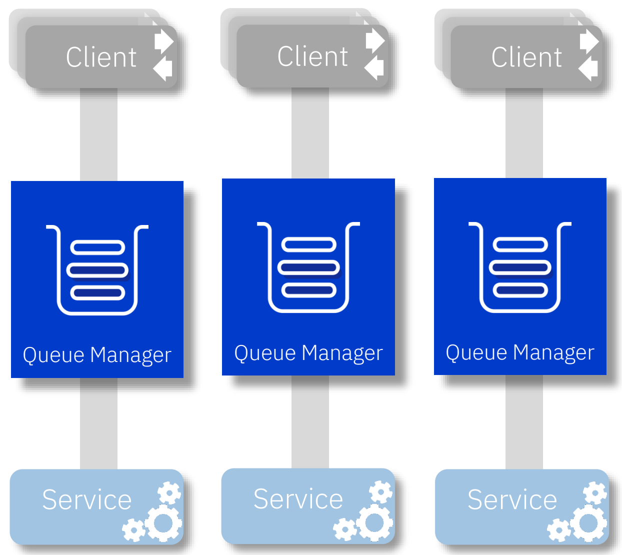
Gateway routing



Q. Is clustering required?

A. Definitely

Scaled out applications



Q. Is clustering required?

A. Maybe, maybe not ...

© 2019 IBM Corporation

Back to the *uniform cluster*

Building scalable, fault tolerant, solutions

Many of you have built your own continuously available and horizontally scalable solutions over the years

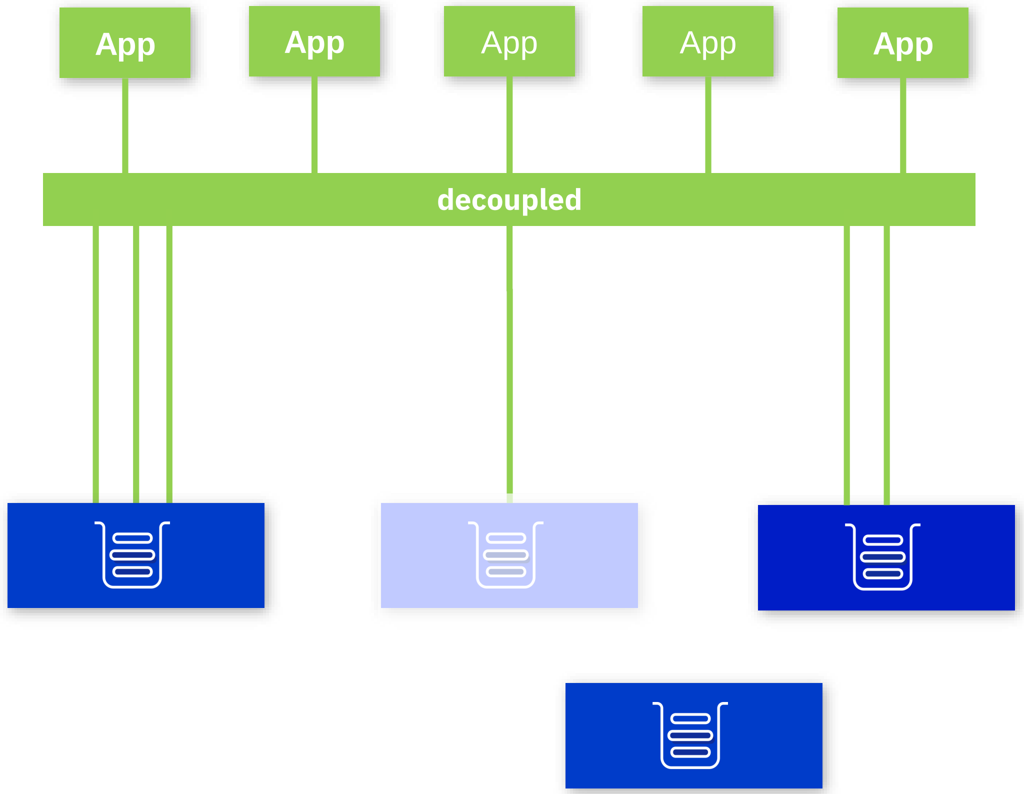
Let’s call this the *“uniform cluster”* pattern

MQ has provided you many of the building blocks -

- Client auto-reconnect
- CCDT queue manager groups

But you’re left to solve some of the problems, particularly with long running applications -

- Efficiently distributing your applications
- Ensuring all messages are processed
- Maintaining availability during maintenance
- Handling growth and contraction of scale



© 2019 IBM Corporation

MQ 9.1.2 is starting to make that easier

IBM MQ 9.1.2 CD

For the distributed platforms, declare a set of matching queue managers to be following the uniform cluster pattern

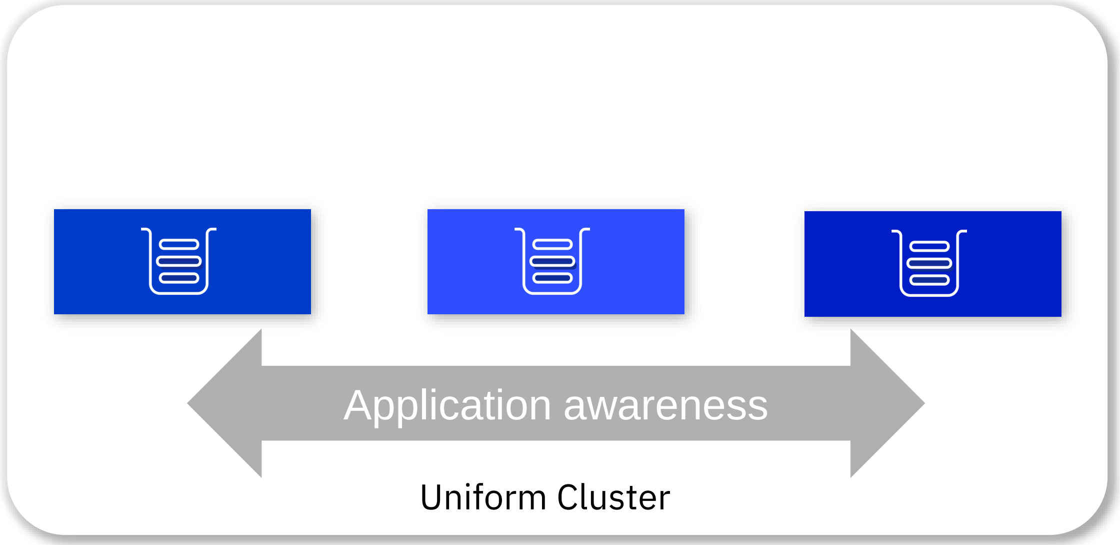
- All members of an MQ Cluster
- Matching queues are defined on every queue manager
- Applications can connect as clients to every queue manager

MQ will automatically share application connectivity knowledge between queue managers

The group will use this knowledge to automatically keep matching application instances balanced across the queue managers

- Matching applications are based on application name (new abilities to programmatically define this)

MQ 9.1.2 is starting to roll out the client support for this



https://developer.ibm.com/messaging/2019/03/21/building-scalable-fault-tolerant-ibm-mq-systems/

© 2019 IBM Corporation

Automatic Application balancing

IBM MQ 9.1.2 CD

Application instances can initially connect to any member of the group

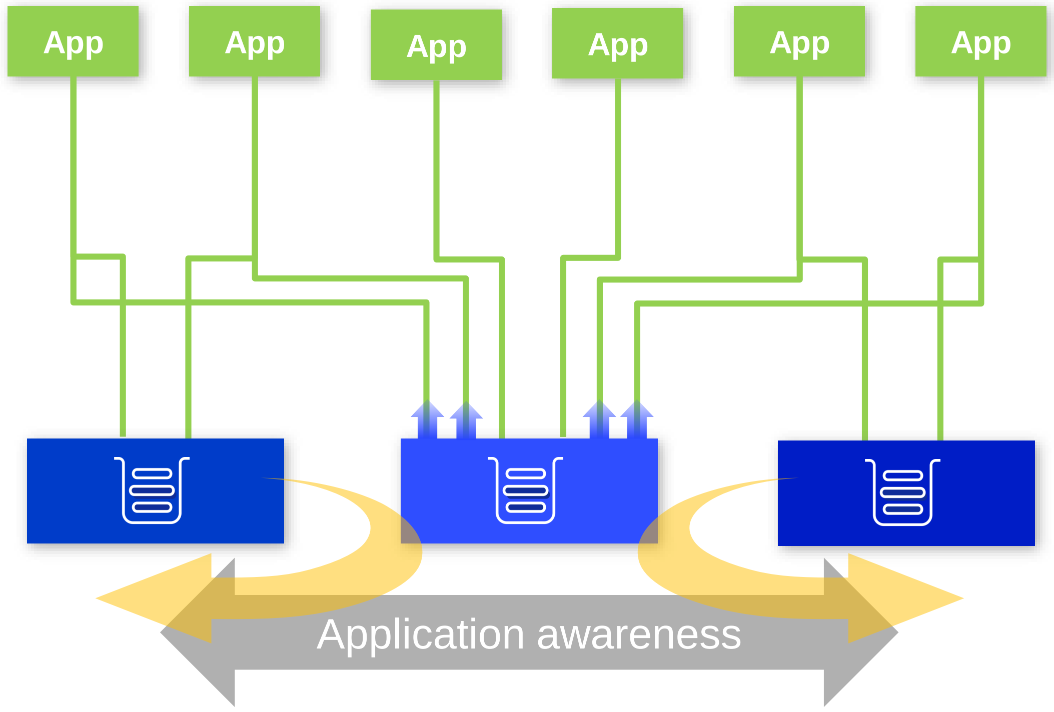
- We recommend you use a queue manager group and CCDT to remove any SPoF

Every member of the uniform cluster will detect an imbalance and request other queue managers to donate their applications

Hosting queue managers will instigate a client auto-reconnect with instructions of where to reconnect to

Applications that have enabled auto-reconnect will automatically move their connection to the indicated queue manager

- 9.1.2 CD has started with support for C-based applications …



https://developer.ibm.com/messaging/2019/03/21/building-scalable-fault-tolerant-ibm-mq-systems/

© 2019 IBM Corporation

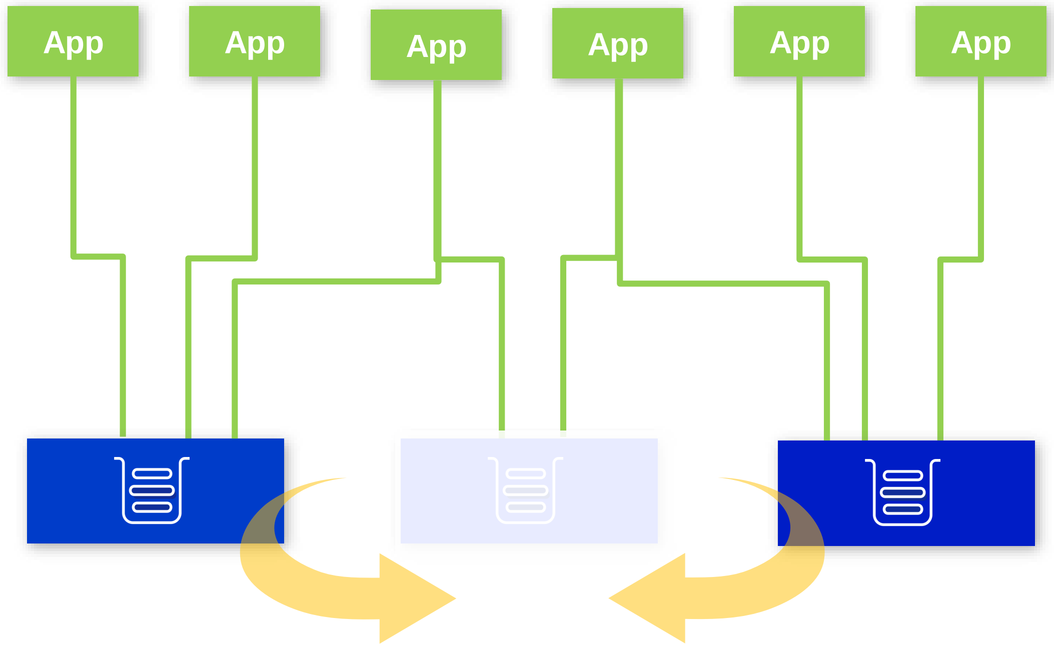
Automatic Application balancing

IBM MQ 9.1.2 CD

Automatically handle rebalancing following planned and unplanned queue manager outages

Existing client auto-reconnect and CCDT queue manager groups will enable initial re-connection on failure

Uniform Cluster rebalancing will enable automatic rebalancing on recovery



https://developer.ibm.com/messaging/2019/03/21/building-scalable-fault-tolerant-ibm-mq-systems/

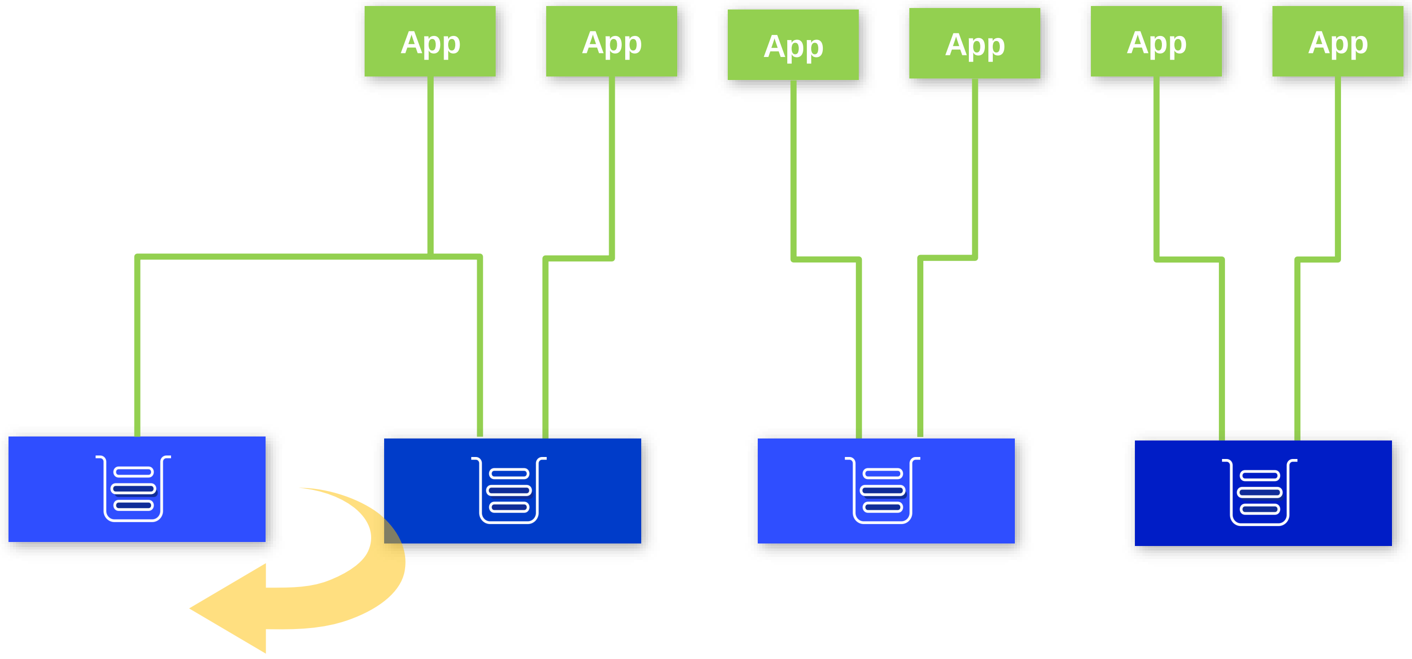
© 2019 IBM Corporation

Automatic Application balancing

IBM MQ 9.1.2 CD

Even to horizontally scale out a queue manager deployment

- Simply add a new queue manager to the uniform cluster
- The new queue manager will detect an imbalance of applications and request its fair share



https://developer.ibm.com/messaging/2019/03/21/building-scalable-fault-tolerant-ibm-mq-systems/

© 2019 IBM Corporation

Uniform Cluster features

IBM MQ 9.1.2 CD

As well as the automatic rebalancing of the C library based clients, MQ 9.1.2 CD introduces a number of new or improved features for the distributed platforms that tie together to make all this possible

This means you need 9.1.2 both for the queue managers *and* the clients

Creation of a Uniform Cluster

- A simple qm.ini tuning parameter for now

```
TuningParameters:
   UniformClusterName=CLUSTER1
```

The ability to identify applications by name, to define grouping of related applications for balancing

- Extends the existing JMS capability to all languages

```
$ export MQAPPLNAME=MY.SAMPLE.APP
```

Auto reconnectable applications

- Only applications that connect with the auto-reconnect option are eligible for rebalancing

```
Channels:
   DefRecon=YES
```

Text based CCDTs to make it easier to configure this behaviour

- And to allow duplicate channel names

```
...
```

https://developer.ibm.com/messaging/2019/03/21/walkthrough-auto-application-rebalancing-using-the-uniform-cluster-pattern/

© 2019 IBM Corporation

Balancing by application name

IBM MQ 9.1.2 CD

Automatic application balancing is based on the application name alone

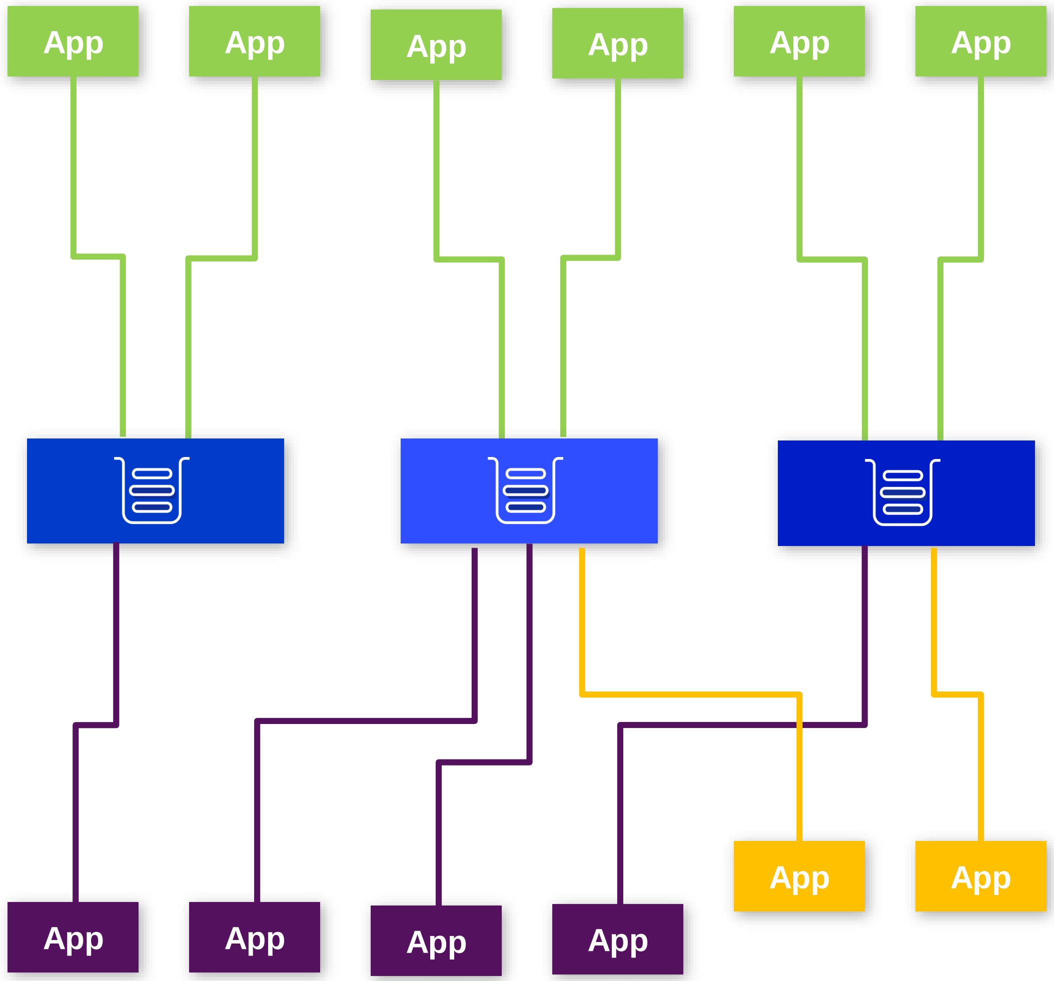
Different groups of application instances with different application names are balanced independently

By default the application name is the executable name

This has been customisable with Java and JMS applications for a while

MQ 9.1.2 CD clients have extended this to other programming languages
For example C, .NET, XMS, …

Application name can be set either programmatically or as an environment override



https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.dev.doc/q132920_.htm

© 2019 IBM Corporation

Building scalable and available solutions

IBM MQ 9.1.2 CD

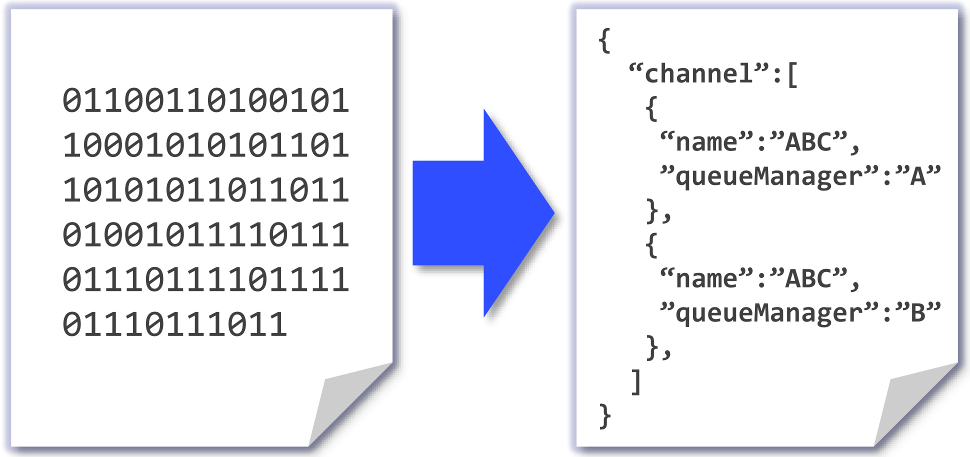
JSON CCDT

Build your own JSON format CCDTs

Supports multiple channels of the same name on different queue managers to simplify the building of uniform clusters

Available with all 9.1.2 clients

- C, JMS, .NET, Node.js, Golang clients



```
01100110100101
10001010101101
10101011011011
01001011110111
01110111101111
01110111011
```

```
{
  “channel”:[
   {
    “name”:”ABC”,
     ”queueManager”:”A”
   },
   {
    “name”:”ABC”,
     ”queueManager”:”B”
   },
  ]
}
```

© 2019 IBM Corporation

IBM MQ 9.1.2 CD

Configuring the CCDT for application balancing in a Uniform Cluster

To correctly setup a CCDT for application rebalancing it needs to contain **two** entries per queue manager:

- An entry under the name of a ***queue manager group***
- And entry under the queue *manager's **real name***

(These previously would need to be different channels, but with the JSON CCDT this is unnecessary)

The application connects using the queue manager group as the queue manager name (prefixed with an '*')

```
{
  "channel":
  [
    {
      "name": ”SVRCONN.CHANNEL",
      "type": "clientConnection”
      "clientConnection":
      {
        "connection":
        [
          {
            "host": ”host1",
            "port": 1414
          }
        ],
        "queueManager": "ANY_QM”
      },
    },
    {
      "name": ”SVRCONN.CHANNEL",
      "type": "clientConnection”
      "clientConnection":
      {
        "connection":
        [
          {
            "host": ”host2",
            "port": 1414
          }
        ],
        "queueManager": "ANY_QM”
      },
    },
  …
```

```
    …
    {
      "name": ”SVRCONN.CHANNEL",
      "type": "clientConnection”
      "clientConnection":
      {
        "connection":
        [
          {
            "host": ”host1",
            "port": 1414
          }
        ],
        "queueManager": ”QMGR1”
      },
    },
    {
      "name": ”SVRCONN.CHANNEL",
      "type": "clientConnection”
      "clientConnection":
      {
        "connection":
        [
          {
            "host": ”host2",
            "port": 1414
          }
        ],
        "queueManager": ”QMGR2”
      },
    }
  ]
}
```

QMGR1

QMGR2

© 2019 IBM Corporation

Can I decouple any application?

Does this work for all applications? – no

This pattern of loosely coupled applications only works for certain applications styles.

Good

Applications that can tolerate being moved from one queue manager to another without realising and can run with multiple instances

- Datagram producers and consumers
- Responders to requests, e.g. MDBs
- No message ordering

Bad

Applications that create persistent state across multiple messaging operations, or require a single instance to be running

- Requestors waiting for specific replies
- Dependant on message ordering
- Global transactions ...

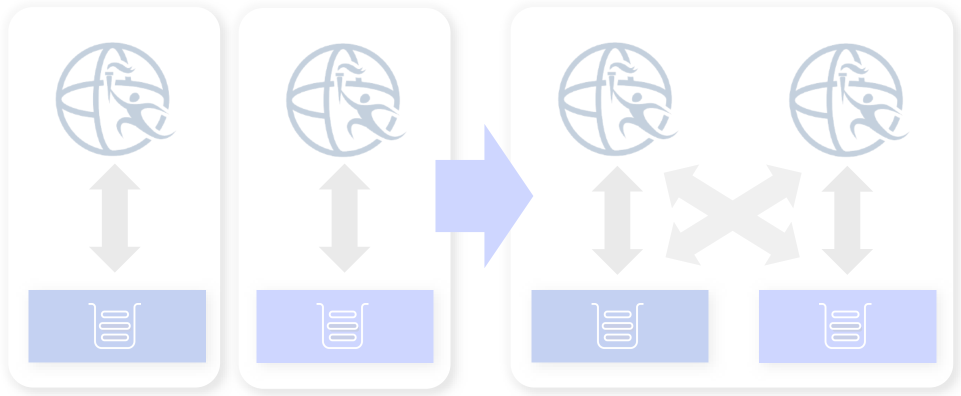
© 2019 IBM Corporation

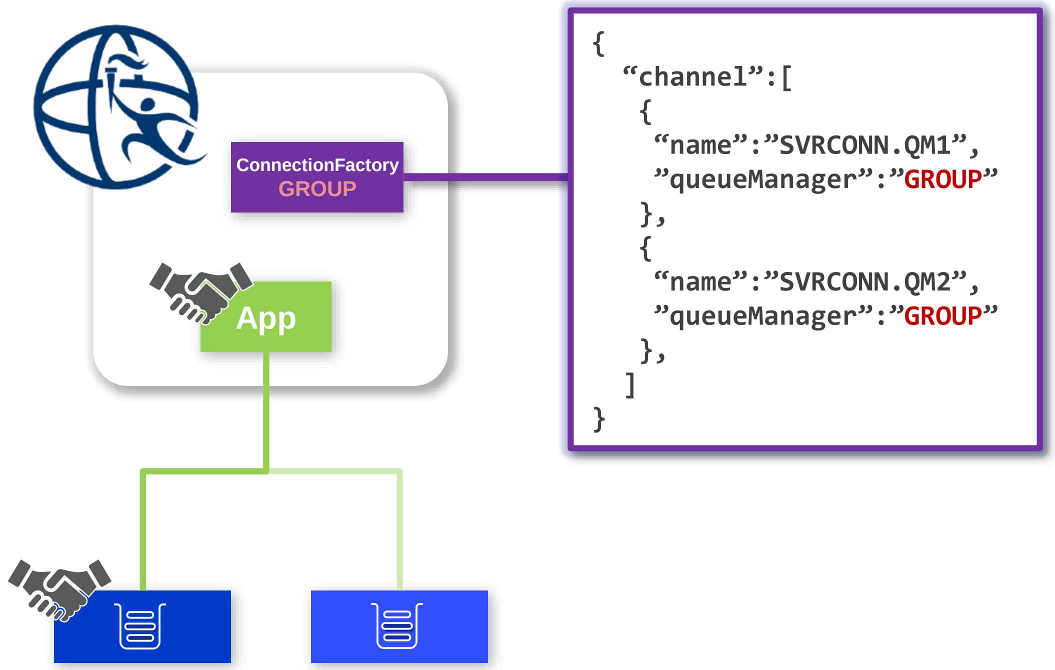
A new hope for transactions

Global transactions require a single resource manager to be named when connecting. For MQ a resource manager is a queue manager.

This prevents the use of queue manager groups in CCDTs

However, **WebSphere Liberty 18.0.0.2** and **MQ 9.1.2 CD** support the use of CCDT queue manager groups when connecting





```
{
  "channel":[
   {
    "name":"SVRCONN.QM1",
    "queueManager":"GROUP"
   },
   {
    "name":"SVRCONN.QM2",
    "queueManager":"GROUP"
   },
  ]
}
```

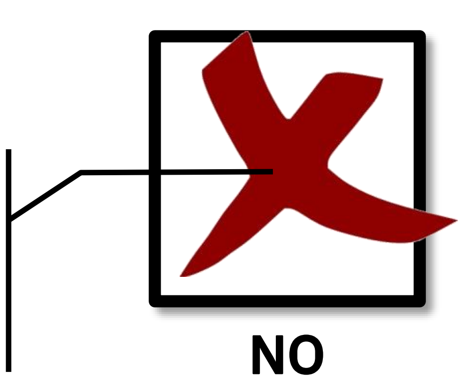
© 2019 IBM Corporation

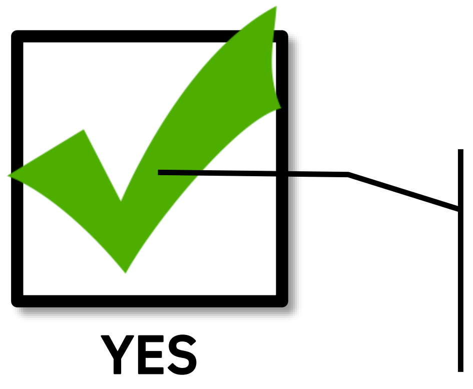
Availability routing in an MQ Cluster

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.

© 2019 IBM Corporation

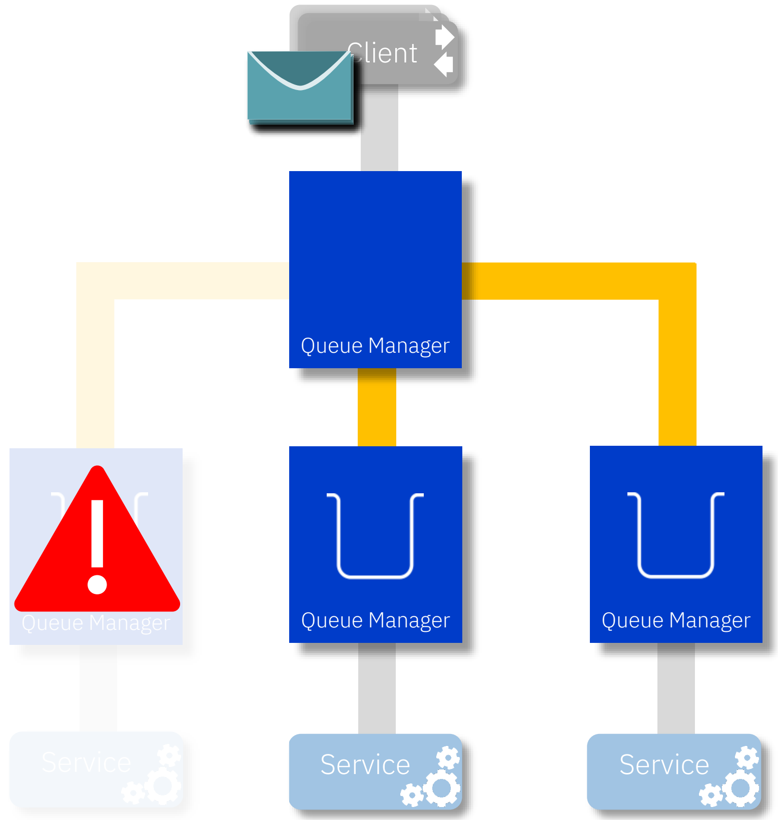
Channel availability routing

- 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



© 2019 IBM Corporation

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

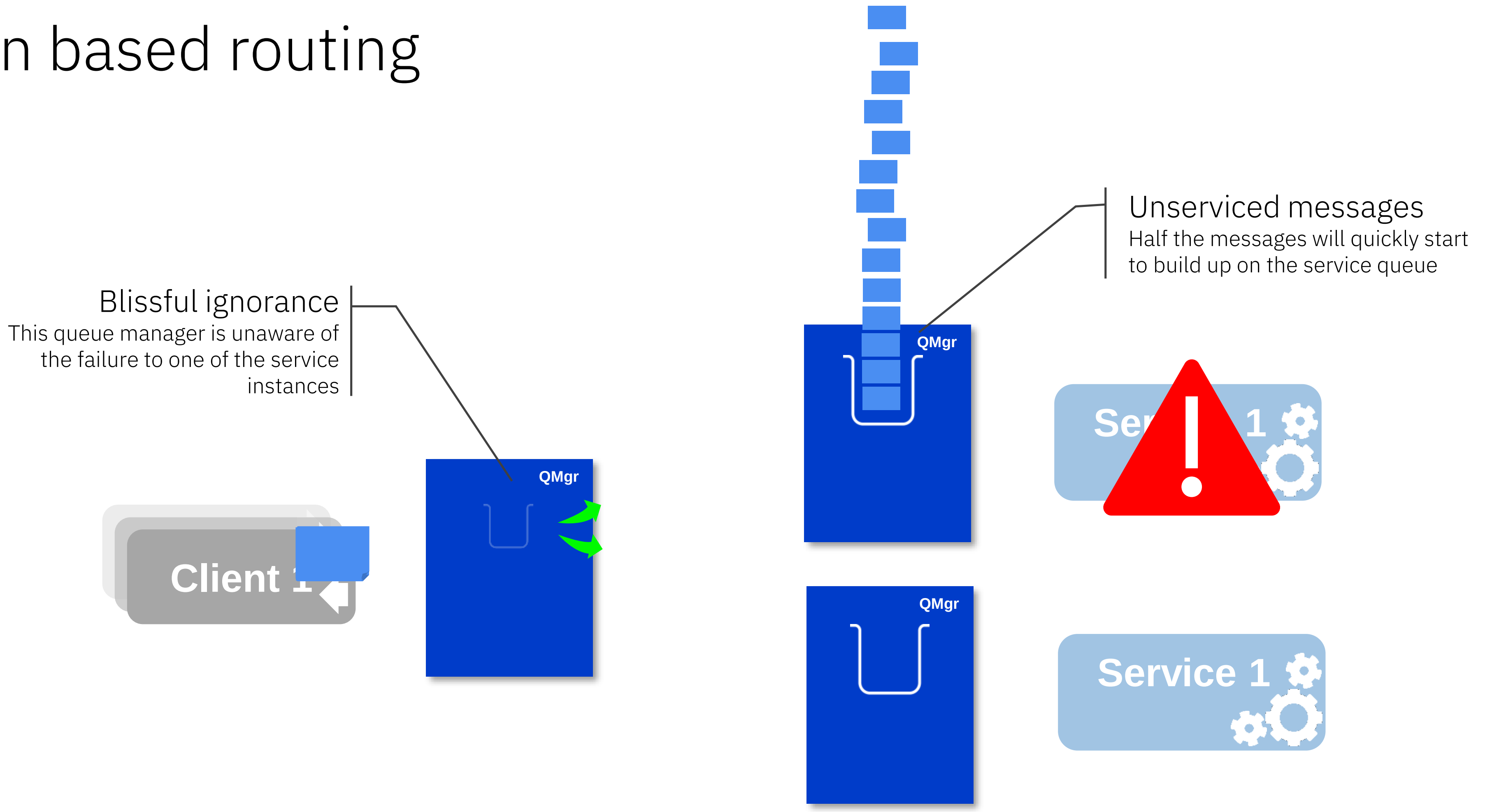
Bind on open is the default
It could be set on the cluster queue (don't forget aliases) or in the app

* 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)

© 2019 IBM Corporation

Application availability routing

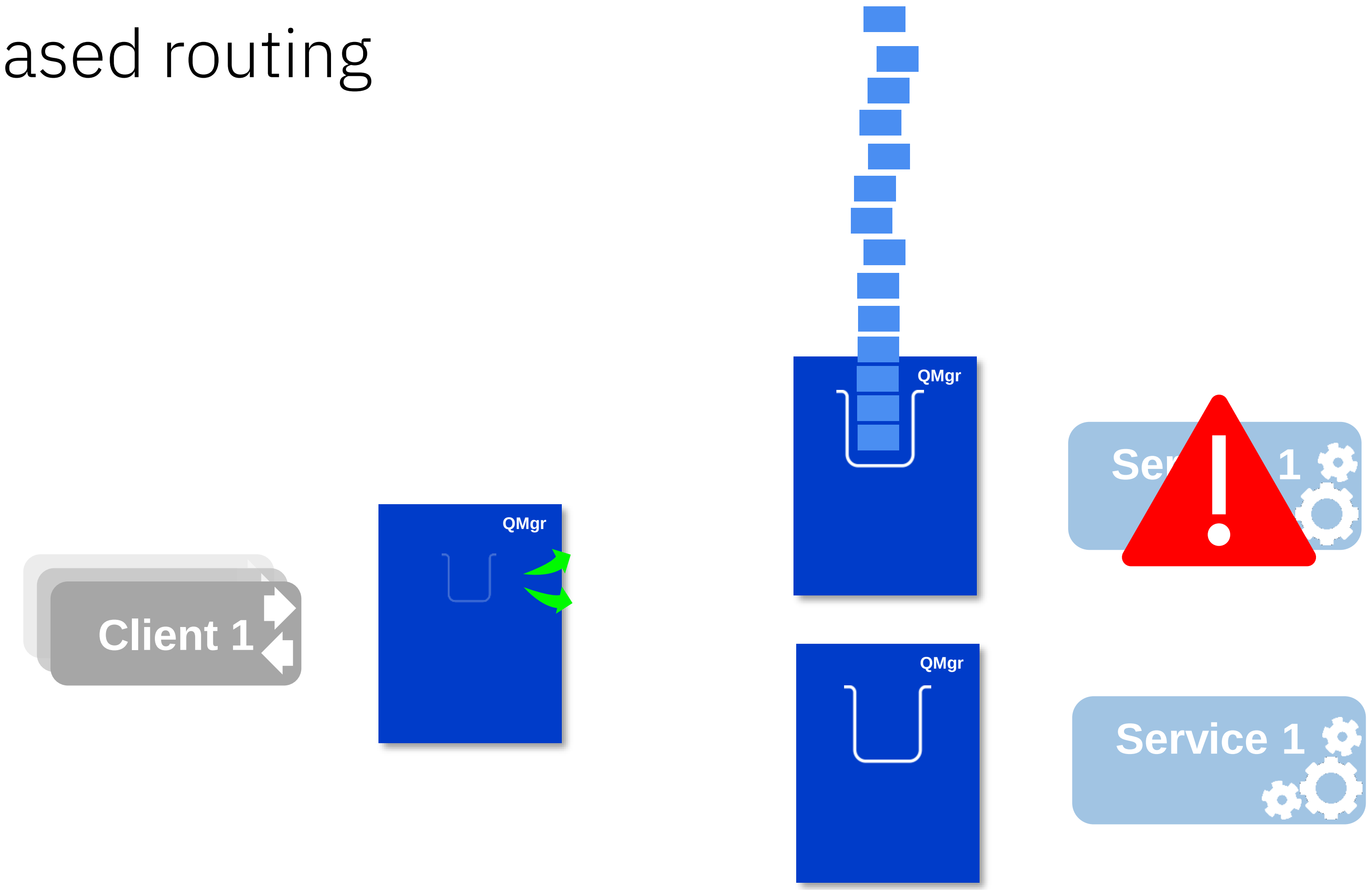
Application based routing


Unserviced messages
Half the messages will quickly start to build up on the service queue
Blissful ignorance
This queue manager is unaware of the failure to one of the service instances
QMgr
QMgr
QMgr
Client 1
Ser 1
Service 1

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

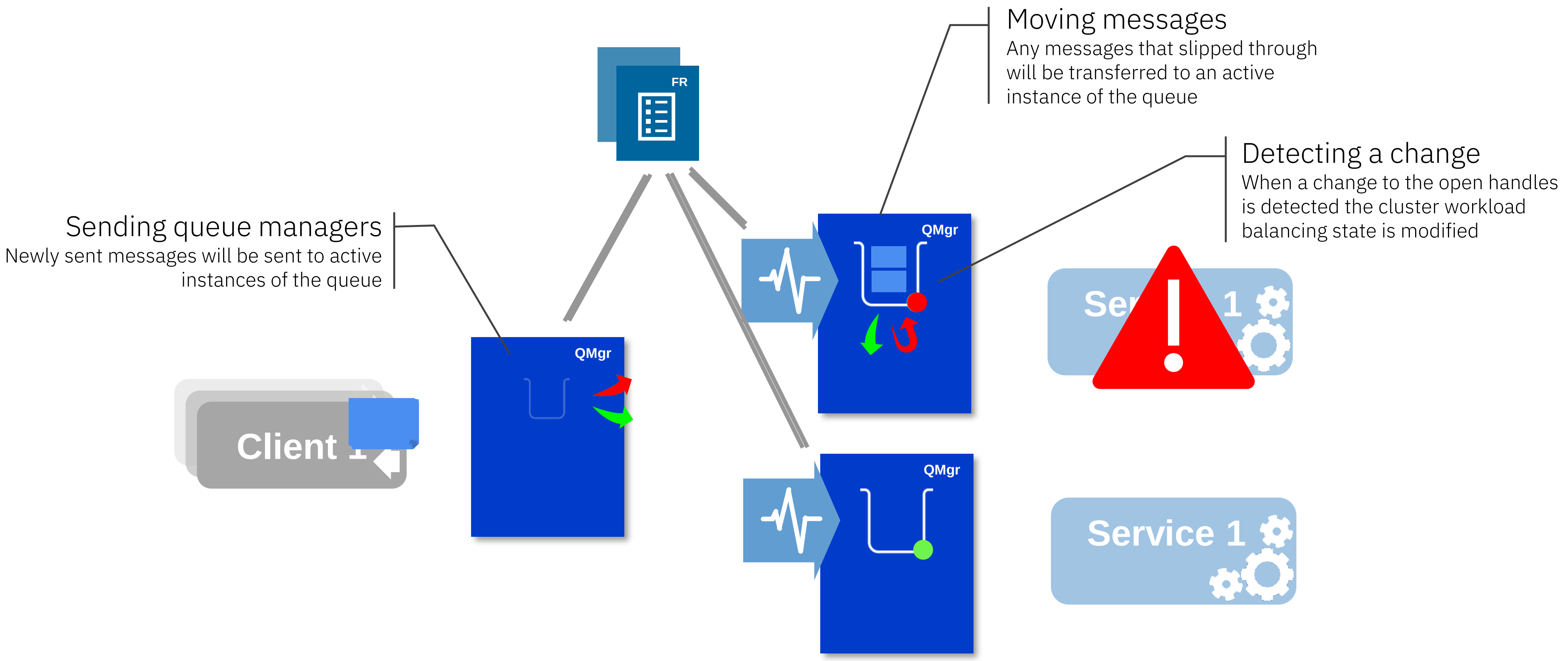
© 2019 IBM Corporation

Application based routing



© 2019 IBM Corporation

Application based routing


Moving messages
Any messages that slipped through will be transferred to an active instance of the queue
Detecting a change
When a change to the open handles is detected the cluster workload balancing state is modified
Sending queue managers
Newly sent messages will be sent to active instances of the queue
FR
QMgr
QMgr
QMgr
Client 1
Service 1
Service 1

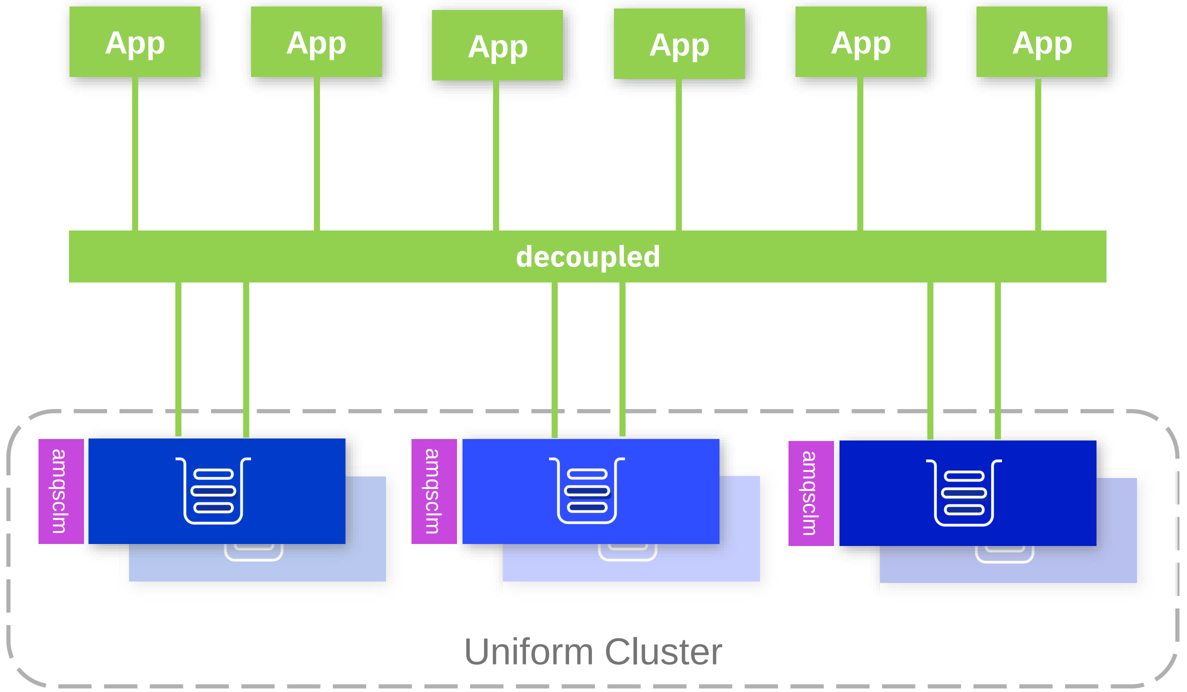
- MQ provides a sample monitoring service tool, **amqsclm**
- It regularly checks for attached consuming applications (IPPROCS)
- And automatically adjusts the cluster queue definitions to route messages intelligently (CLWLPRTY)
- That information is automatically distributed around the cluster

© 2019 IBM Corporation

Putting it all together

Bringing it all together

- Build a matching set of queue managers, in the *style* of a uniform cluster
- Make them highly available to prevent stuck messages
- Consider adding amqsclm to handle a lack of consumers
- Setup your CCDTs for decoupling applications from individual queue managers
- Look at the 9.1.2 application rebalancing capability and see if it matches your needs
- Connect your applications



© 2019 IBM Corporation

Thank You

David Ware
Chief Architect, IBM MQ
dware@uk.ibm.com
www.linkedin.com/in/dware1