Integration Technical Conference 2019

Session M04
MQ High Availability

Your message data is critical,
MQ HA makes sure it's available

Anthony Beardsmore
IBM MQ Development

IBM Cloud

High availability

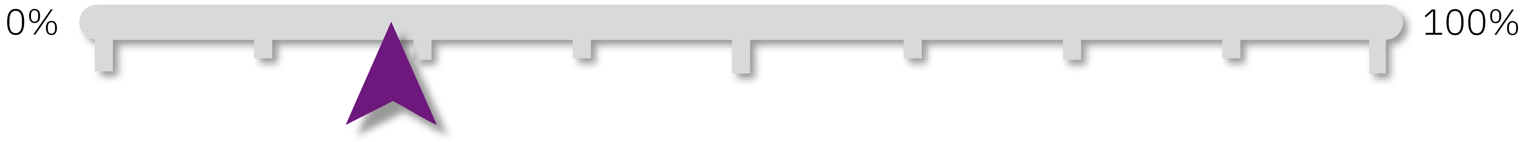
Available

A system is said to be available if it is able to perform its required function, such as successfully process requests from users.

Highly Available

A requirement, or a capability, of a system to be operational for a greater proportion of time than is common for other, less important, systems.

A sliding scale



Often, greater availability means greater complexity and cost

Measuring availability

Target	Yearly outage
95%	~ 18 days
99%	< 4 days
99.5%	< 2 days
99.9%	< 9 hours
99.99%	1 hour
99.999%	~ 5 minutes
99.9999%	~ 30 seconds

‘5 nines’

Impacts on availability

- Applying maintenance
- Likelihood of outages (meantime to failure, and speed of recovery)
- Operational errors

Overall availability is the combined availability of all components

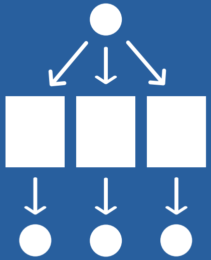
- The platform
- The middleware
- The applications

Messaging system availability

Asynchronous messaging can improve application availability by providing a buffer but the messaging system itself must be highly available to achieve that

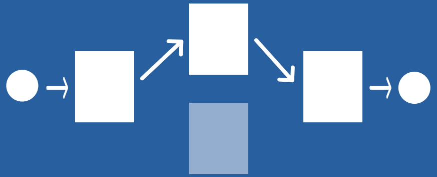
Redundancy

Multiple active options available for applications to connect



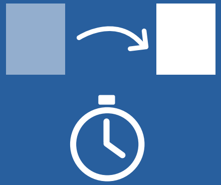
Routing

Ability to route messages around failures



Message availability

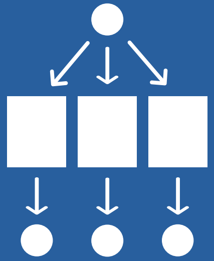
Critical messages are not locked to a single runtime and quickly available from elsewhere



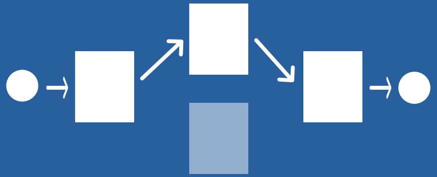
Messaging system availability

Asynchronous messaging can improve application availability by providing a buffer but the messaging system itself must be highly available to achieve that

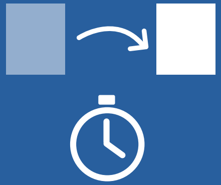
Redundancy
Multiple active options available for applications to connect



Routing
Ability to route messages around failures



Message availability
Critical messages are not locked to a single runtime and quickly available from elsewhere



Required for highest system availabilities

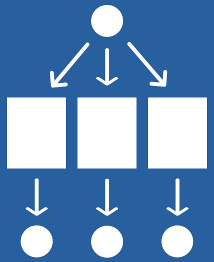
Only required for certain message flows

Messaging system availability – **MQ**

Asynchronous messaging can improve application availability by providing a buffer but the messaging system itself must be highly available to achieve that

Redundancy

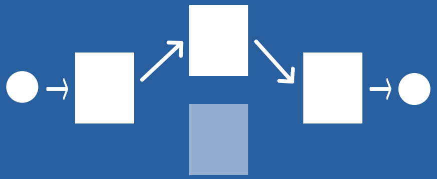
Multiple active options available for applications to connect



Application client connectivity

Routing

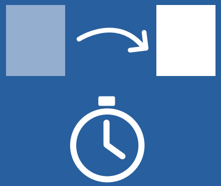
Ability to route messages around failures



MQ Cluster routing

Message availability

Critical messages are not locked to a single runtime and quickly available from elsewhere



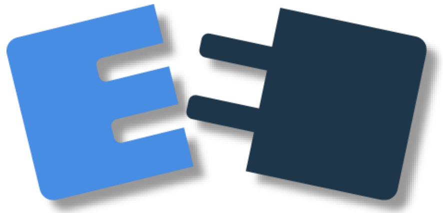
MQ 'HA'

Application client connectivity

Decouple the applications from queue managers

Applications locally bound to a queue manager will limit the availability of the solution.

Running applications remote from the queue managers, always connecting as MQ clients, decouples the application and system runtimes, enabling higher availability.



Decouple the applications from queue managers

Step 1

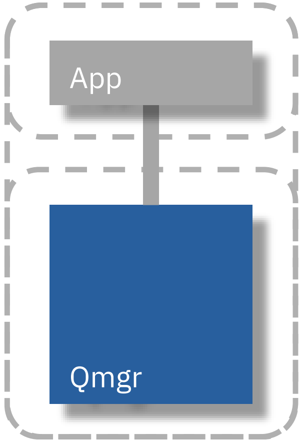
Connect the application as a client

Benefits:

- Ability to support solutions where a queue manager may fail-over between systems (more later).
- Separates application system requirements from the queue manager's, reducing maintenance conflicts and therefore, availability.
- Restart times on either side can be reduced.

Should be relatively invisible to the application

- Don't hardcode that connection configuration!
- Use **client auto-reconnect** to hide a queue manager restart from the application



Decouple the applications from queue managers

Step 2

Allow the application to connect to a set of queue managers

Benefits:

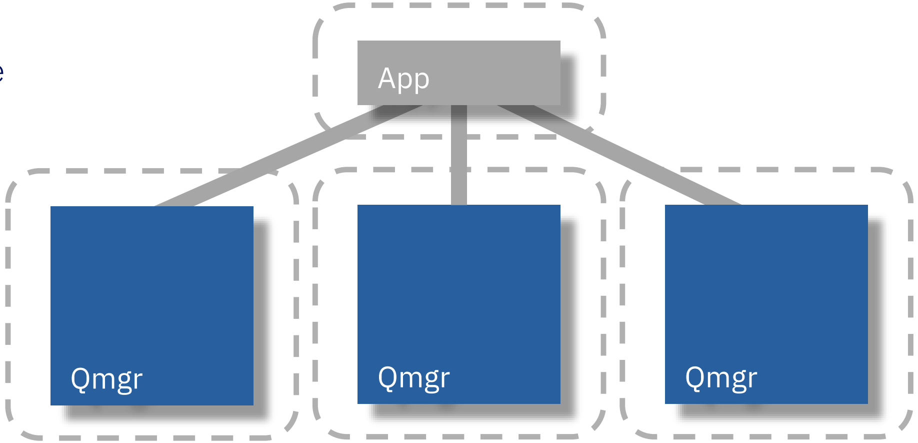
- Applications can continue to interact with MQ even whilst a queue manager is failing over or unavailable during maintenance
- With multiple applications connected, only a subset will be impacted by a queue manager outage

How does your application find the queue manager?

- Network routing
- Connection name lists
- Client Channel Definition Tables

The application may need to re-evaluate how it exploits all of MQ's capabilities

- Message ordering may change if it is currently expected across connections
- Applications may be reliant on transitory state:
  - Dynamic queues and subscriptions
  - Reply messages
  - XA transaction recovery



This might not work for all applications

What do CCDTs enable?

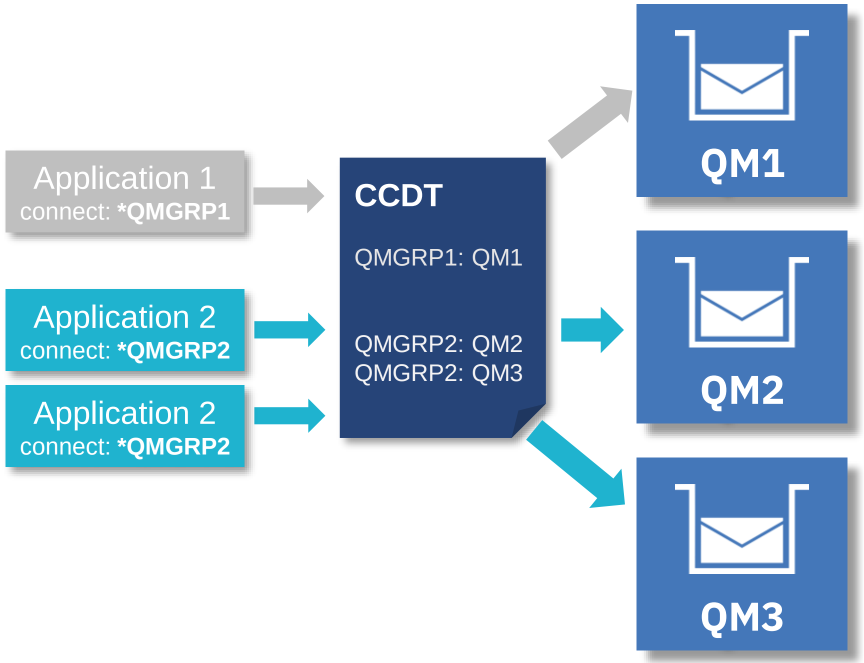
These provide encapsulation and abstraction of connection information for applications, hiding the MQ architecture and configuration from the application

They also enable security, high availability and workload balancing of clients

Applications simply connect to an abstracted “queue manager” name (which doesn’t need to be the actual queue manager name – use an ‘*’)

CCDT defines which real queue managers the application will connect to. Which could be a single queue manager or a group of

Across a group, selection can be ordered or randomised and weighted



Creating the CCDTs

CCDTs can represent connection details to multiple queue managers

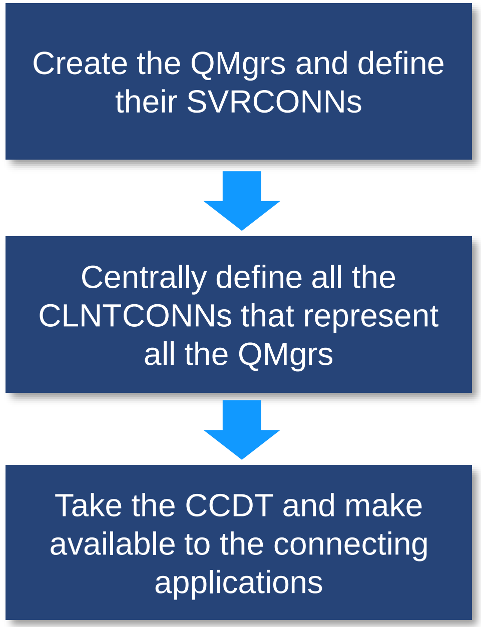
CLNTCONN channels are defined to identify the **SVRCONN** channels

Define multiple CLNTCONNs in a central place to generate the CCDT

- It doesn’t have to be any of the queue managers owning the SVRCONNs
- **Pre-MQ V8:** You needed a dedicated queue manager for this purpose
- **MQ V8+:** Use **runmqsc –n** to remove the need for a queue manager

A **single CCDT** for your MQ estate or **one per application**?

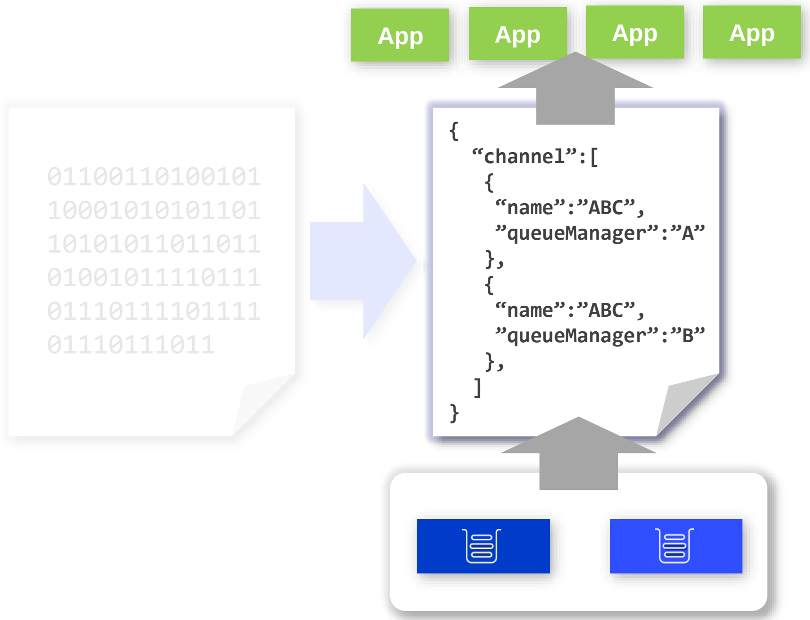
- A single CCDT can be easier to create but updates can be expensive
- Separate CCDTs make it easier to update when an application’s needs change


Create the QMgrs and define their SVRCONNs
Centrally define all the CLNTCONNs that represent all the QMgrs
Take the CCDT and make available to the connecting applications

Creating the CCDTs
New in 9.1.2 (CD)

JSON CCDT

- Build your **own JSON format CCDTs**
- Supports multiple channels of the same name on different queue managers. (More on this is a moment…)
- Available with all 9.1.2 clients
  - C, JMS, .NET, Node.js, Golang clients



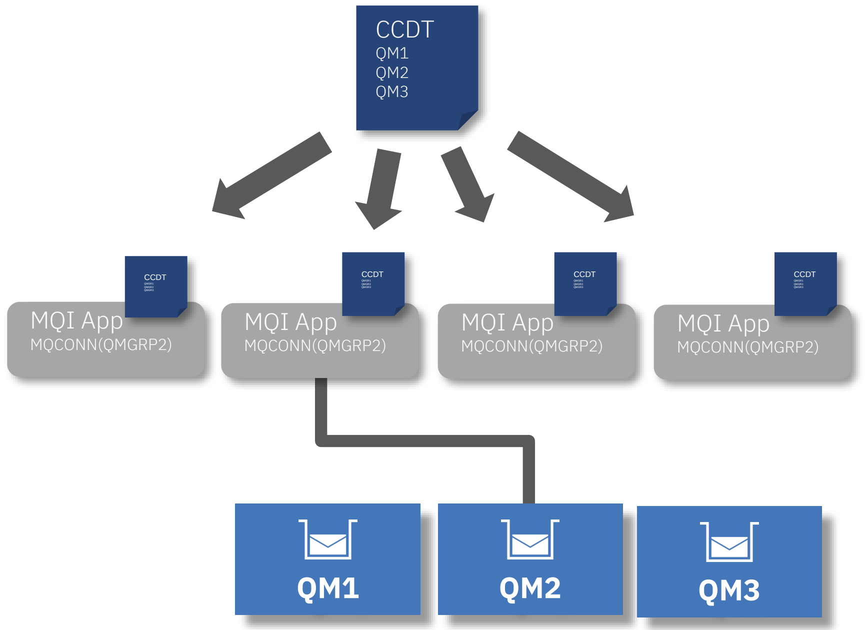
```
{
  "channel":[
   {
    "name":"ABC",
    "queueManager":"A"
   },
   {
    "name":"ABC",
    "queueManager":"B"
   },
  ]
}
```

Accessing the CCDTs

CCDT files need to be accessible to the applications connecting to MQ

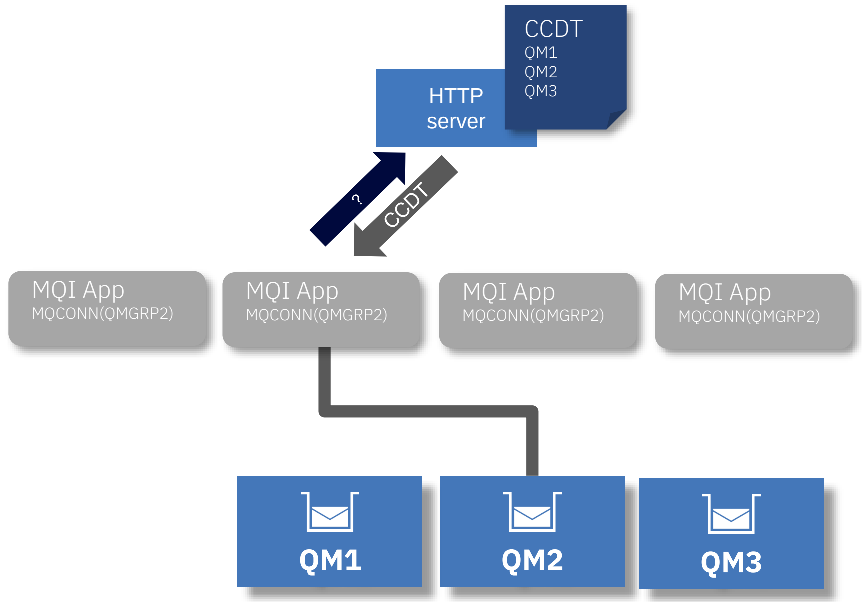
Either accessible through the client's filesystem

User needs to manage distribution of CCDT files themselves

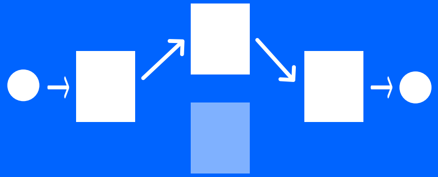


Or remotely over HTTP or FTP

Available for JMS/XMS applications for a number of releases
Added for MQI applications in MQ V9 LTS

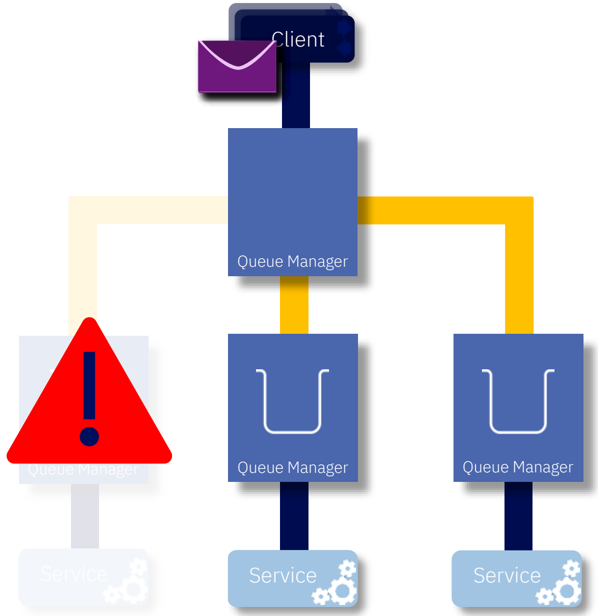


MQ Cluster routing



Routing on availability with MQ Clusters

- MQ Clusters provide a way to route messages based on availability
- In a cluster there can be multiple potential targets for any message. This alone can improve the availability of the solution, always providing an option to process new messages.
- A queue manager in a cluster also has the ability to route new and old messages based on the availability of the channels, routing messages to running queue managers.
- Clustering can also be used to route messages to active consuming applications.
- Clustering is used by many customers who operate critical services at scale
- Available on all supported MQ platforms



Bringing together MQ Clusters and flexible client connectivity

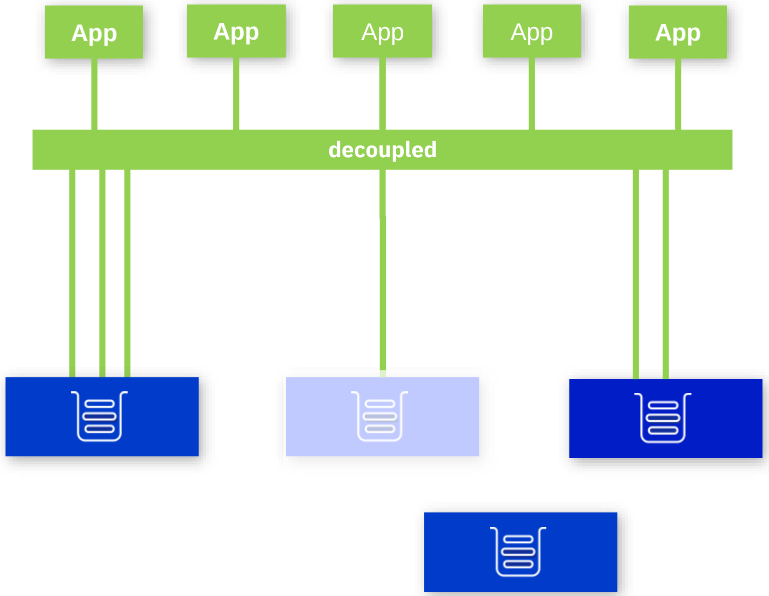
A specific case of MQ Cluster usage is when all queue managers in the cluster provide exactly the same services, and the cluster exists primarily to provide scale and availability to a set of client applications. Lets call this the *__“uniform cluster”__* pattern

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



MQ 9.1.2 is starting to make that easier

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

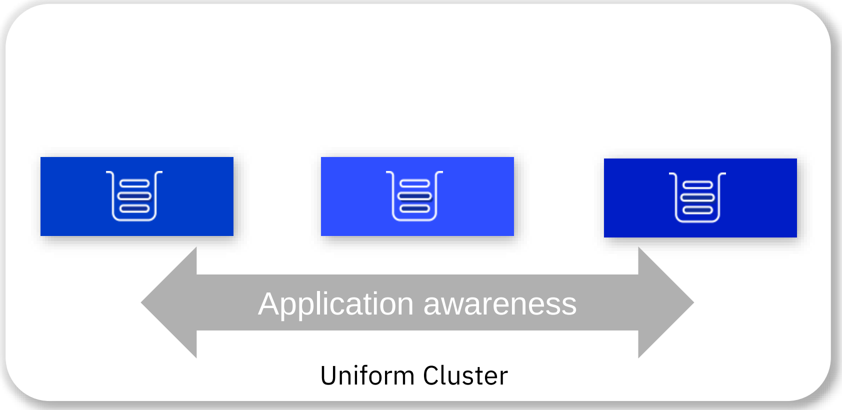
- All members of an MQ Cluster
- Matching queues are defined on every queue manager
- Applications can connect as clients to every queue manager

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 also starting to roll out the client support for this



Automatic Application balancing

Application instances can initially connect to any member of the group

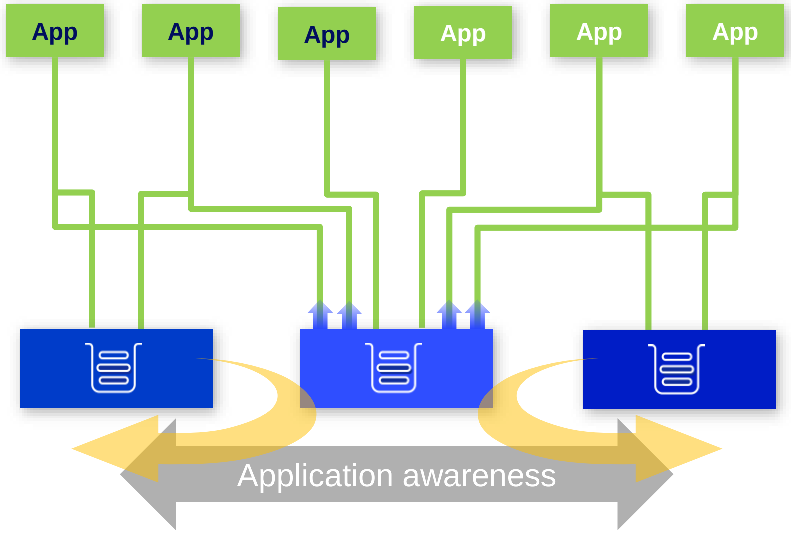
> We recommend you use a queue manager group and CCDT to remove any SPoF

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
> …

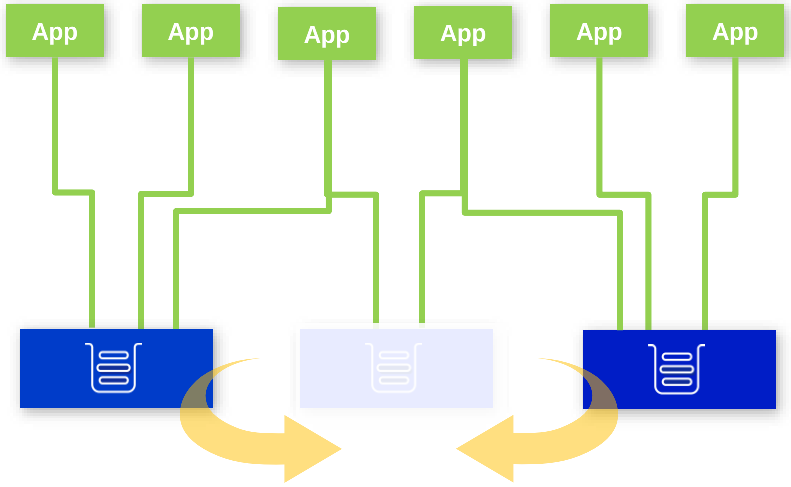


Automatic Application balancing

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

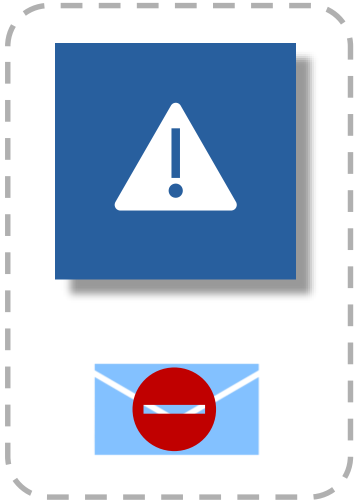
- Uniform Cluster rebalancing will enable automatic rebalancing on recovery



MQ ‘HA’
(message availability)

Message high availability

The problem

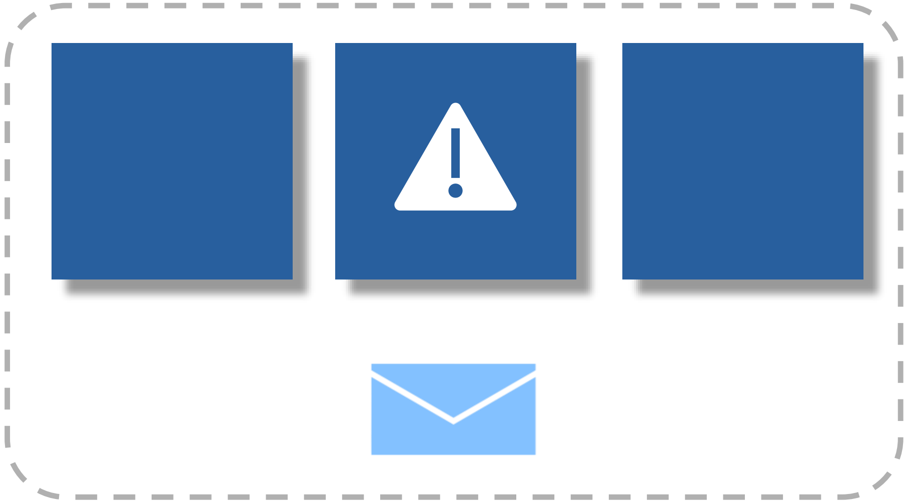


Consider a single message

Tied to a single runtime, on a single piece of hardware

Any failure locks it away until recovery completes

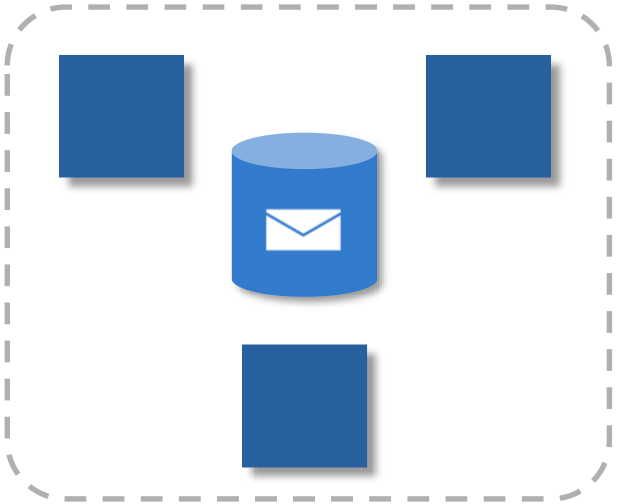
The objective



- Messages are not tied to a single anything
- In the event of a failure, there is a fast route to access the message

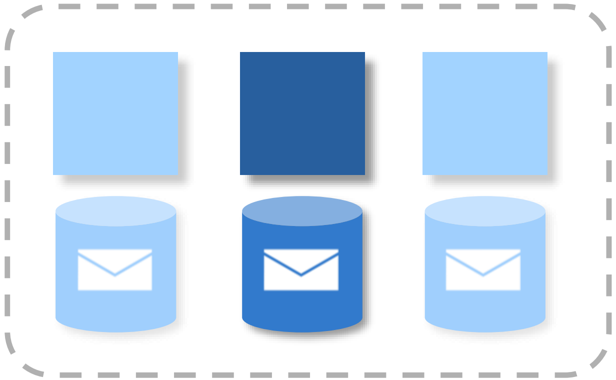
Message high availability

Active / active messages



- Any message is available from any runtime at any time
- Coordinated access to each message
- A failed runtime does not prevent access to a message by another runtime

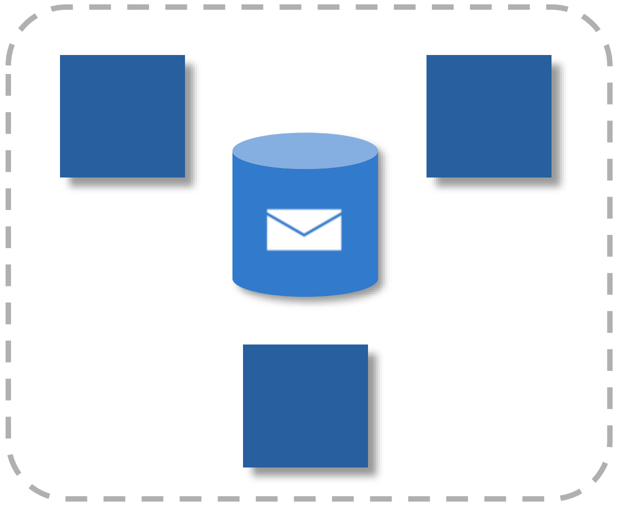
Active / passive messages



- Messages are highly available, through replication
- Only one runtime is the *leader* and has access to the messages at a time
- A failure results in a new leader taking over

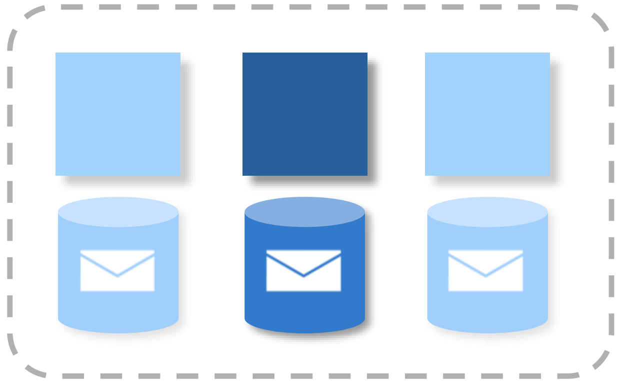
Message high availability

Active / active messages



IBM MQ for z/OS shared queues

Active / passive messages



IBM MQ Distributed HA solutions

MQ for z/OS shared queues

Available with **z/OS parallel sysplex**

– A tightly coupled cluster of independent z/OS instances

Multiple queue managers are members of a **queue sharing group** (QSG)

Shared queues are held in the **Parallel SysPlex Coupling Facility**

– A highly optimised and resilient z/OS technology

All queue managers in a QSG can access the same shared queues and their messages

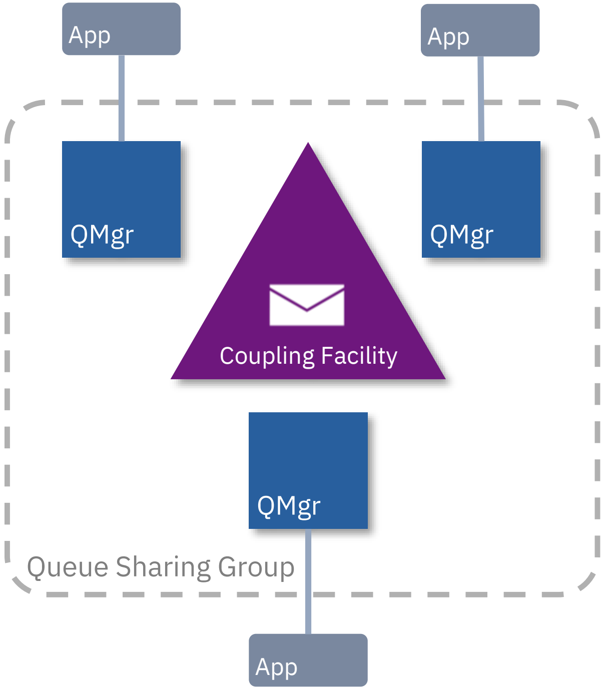
Benefits:

✓ Messages remain available even if a queue manager fails

✓ Pull workload balancing

✓ Applications can connect to the group using a QSG name

✓ Removes affinity to a specific queue manager



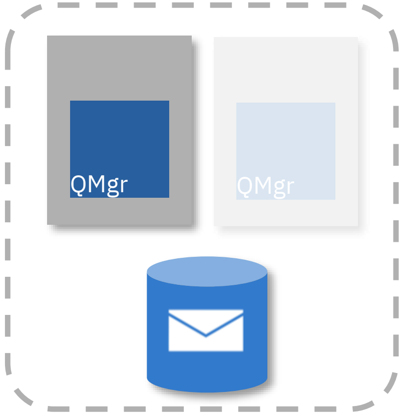
IBM MQ Distributed HA solutions

Externally managed

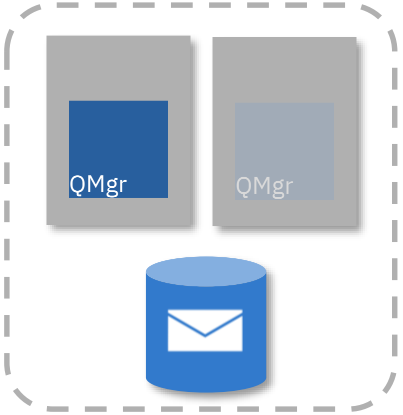
External mechanisms are relied on to protect the data and provide automatic takeover capabilities

MQ managed

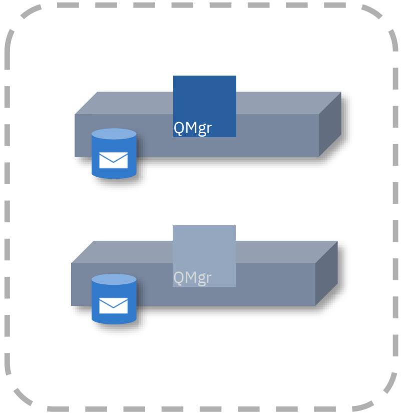
The resilient data and the automatic takeover is provided by the MQ system



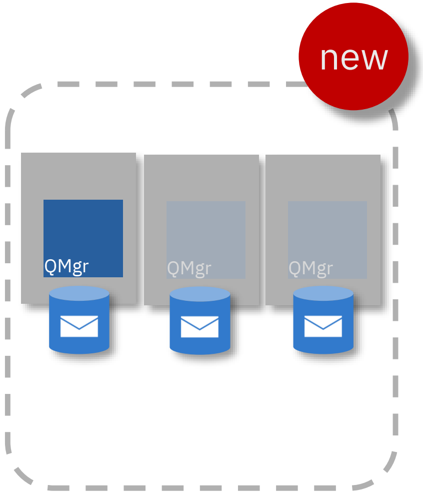
System managed HA



Multi-instance queue managers



MQ Appliance



Replicated data queue managers

Externally managed HA

System managed HA

The HA manager monitors the MQ system (e.g. a queue manager in a VM or container), on detecting a failure it will start a new system, remount storage and reroute network traffic

Relies on external, highly available, storage (e.g. SAN)

A queue manager is unaware of the HA system

Availability depends on speed to detect problems and to restart all layers of the system required (e.g. VM and queue manager)

Examples:

HA Clusters

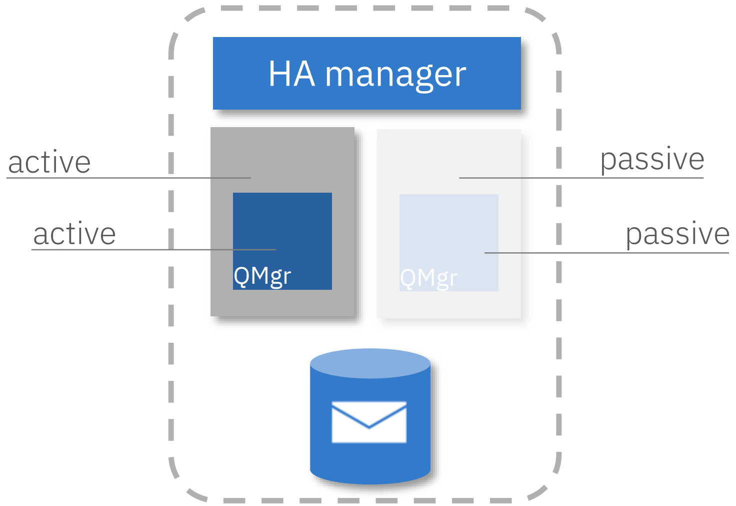
– Veritas Cluster Server, IBM PowerHA, Microsoft Cluster Server

Cloud platforms

– IBM Cloud, AWS EC2, Azure

Containers

– Kubernetes, Docker Swarm



– Some systems can be relatively slow to restart
– Additional cost of infrastructure
– Multiple moving parts to configure and manage

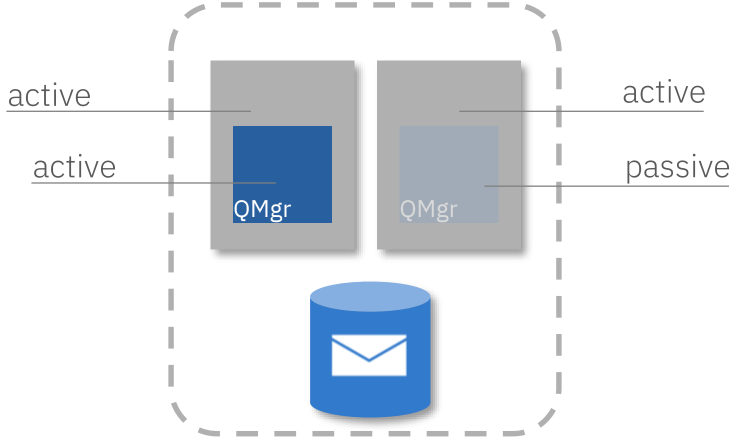
Multi-instance queue managers

All queue manager data is held on **network attached storage** (e.g. NFS, IBM Spectrum Scale).

Two systems are running, both have an instance of the same queue manager, pointed to the same storage. One is **active**, the other is in **standby**.

A failure of the active instance is detected by the standby through regularly attempting to take filesystem locks.

The queue manager with the locks is the active instance.



Faster takeover, less to restart

Cheaper – less specialised software or administration skills needed

Wide platform coverage, Windows, Unix, Linux

- Only as reliable as the network attached storage
- Matching the MQ requirements to filesystem behaviour can be tricky
- No IP address takeover, use client configuration instead

MQ managed HA

IBM MQ Appliance

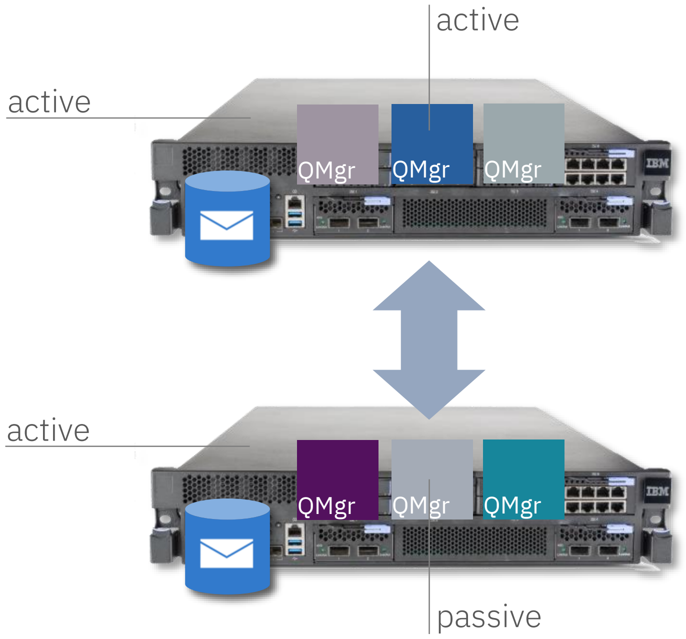
A pair of MQ Appliances are connected together and configured as an HA group

Queue managers created on one appliance can be automatically replicated, along with all the MQ data, to the other

Appliances monitor each other

- Automatic failover, plus manual failover for migration or maintenance
- Independent failover for queue managers so both appliances can run workload (active / active load)
- Optional IP address associated with an HA queue manager, automatically adopted by the active HA appliance – single logical endpoint for client apps

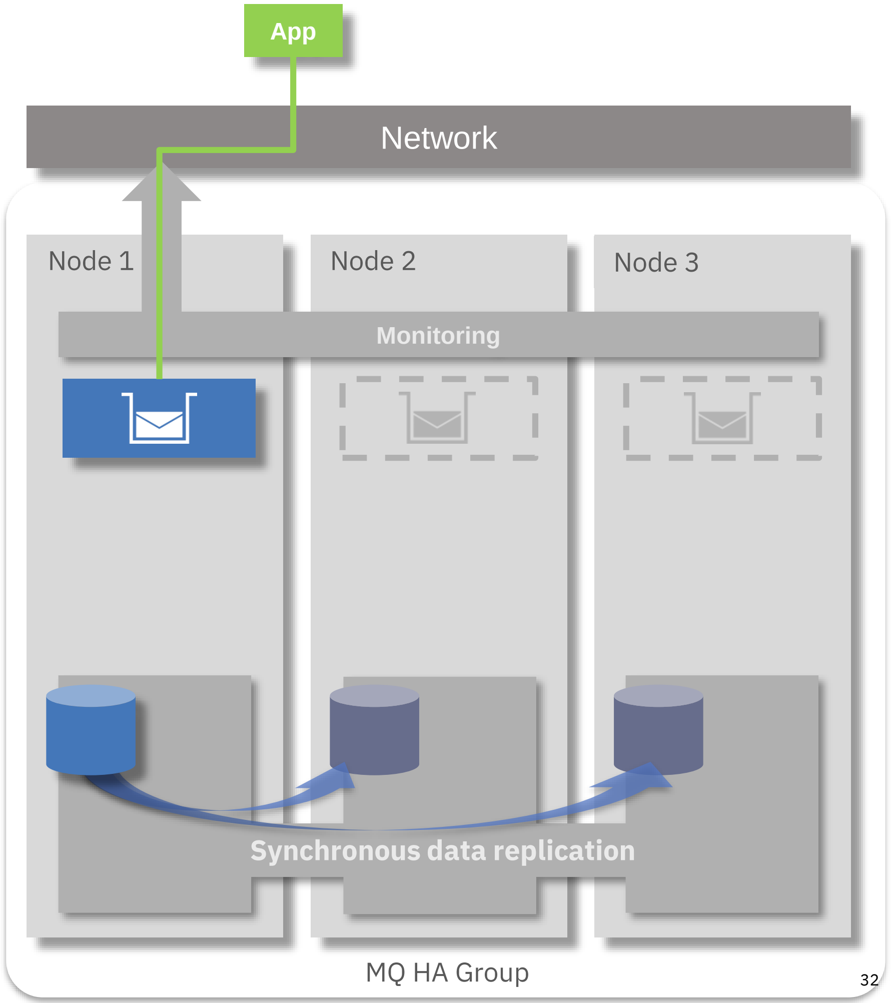
- No persistent data loss on failure
- No external storage
- No additional skills required



Replicated Data Queue Managers

- **Linux only, MQ Advanced** HA solution with no need for a shared file system or HA cluster
- MQ configures the underlying resources to make setup and operations natural to an MQ user
- Three-way replication for quorum support
- **Synchronous** data replication for once and once only transactional delivery of messages
- Active/passive queue managers with **automatic takeover**
- Per queue manager control to support active/active utilisation of nodes
- Per queue manager **IP address** to provide simple application setup
- Supported on **RHEL v7 x86-64 only**

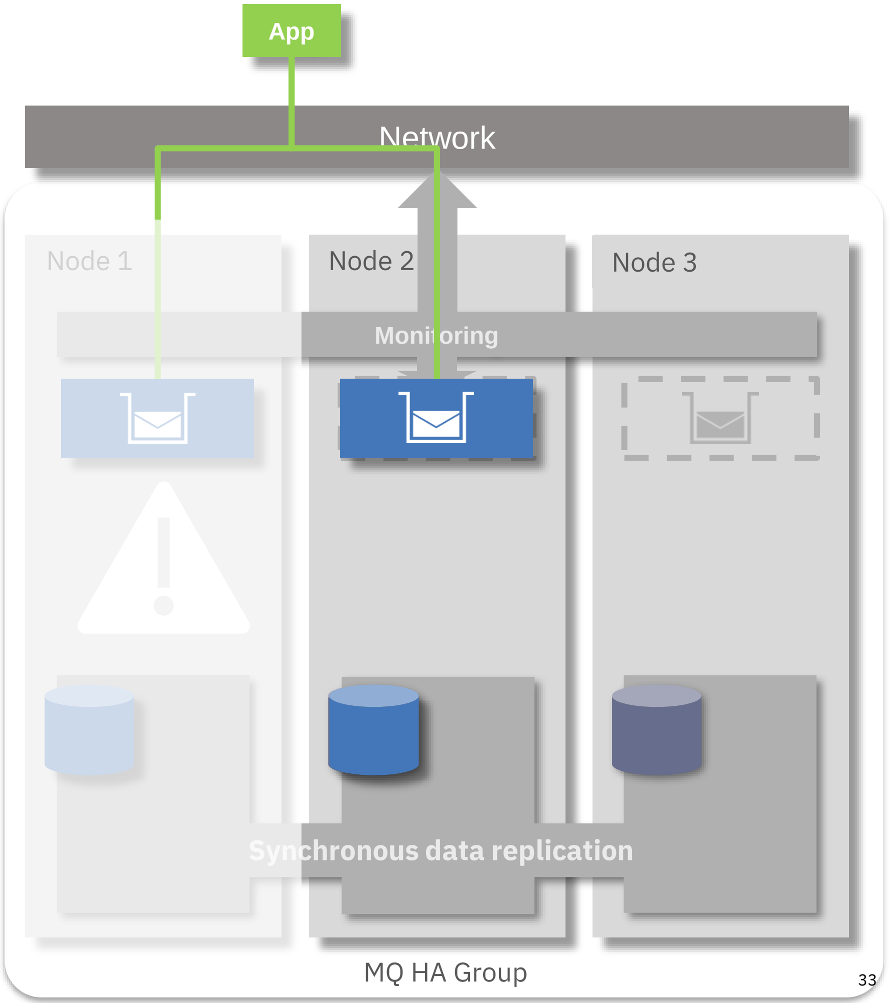
New in V9.0.4 CD / V9.1 LTS MQ Advanced for Linux



Replicated Data Queue Managers

- **Linux only, MQ Advanced** HA solution with no need for a shared file system or HA cluster
- MQ configures the underlying resources to make setup and operations natural to an MQ user
- Three-way replication for quorum support
- **Synchronous** data replication for once and once only transactional delivery of messages
- Active/passive queue managers with **automatic takeover**
- Per queue manager control to support active/active utilisation of nodes
- Per queue manager **IP address** to provide simple application setup
- Supported on **RHEL v7 x86-64 only**

New in V9.0.4 CD / V9.1 LTS MQ Advanced for Linux



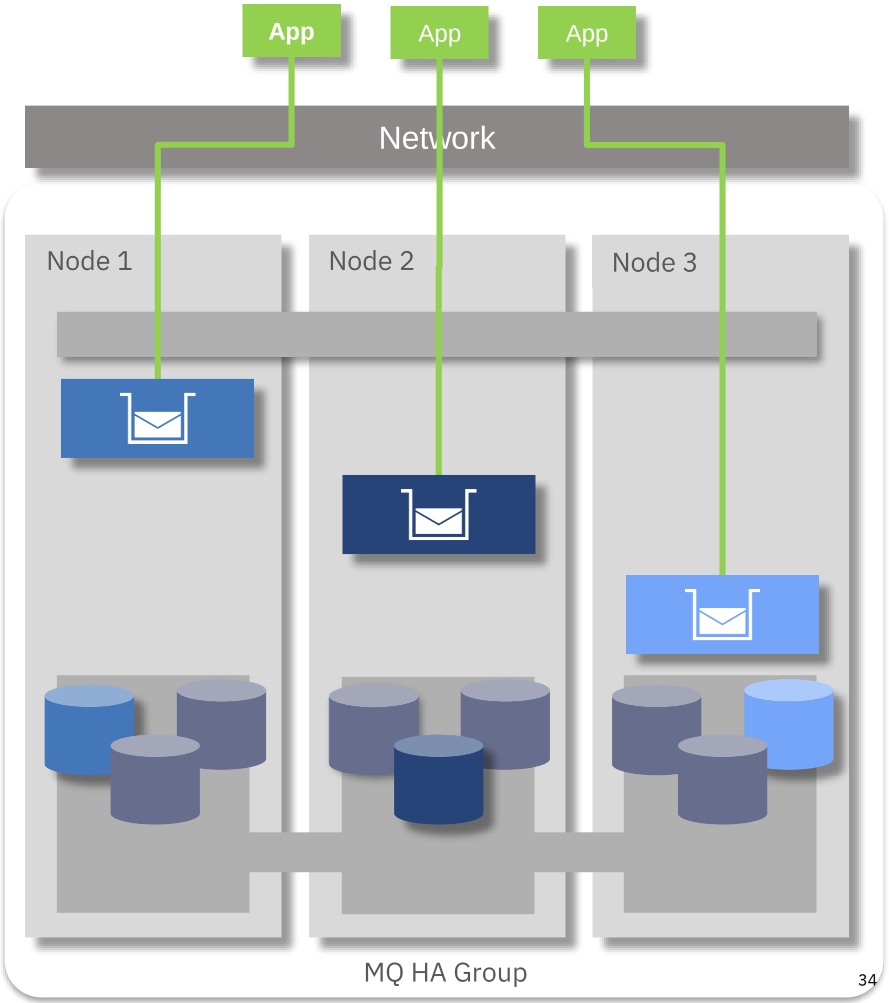
Replicated Data Queue Managers

Recommended deployment pattern:

- Spread the workload across multiple queue managers and distribute them across all three nodes
- Even better, more than one queue manager per node for better failover distribution
- Use MQ Clusters for additional routing of messages to work around problems

MQ **licensing** is aligned to maximise benefits

- **One full IBM MQ Advanced license and two High Availability Replica licenses** (previously named Idle Standby)



External/MQ managed HA with RDQM

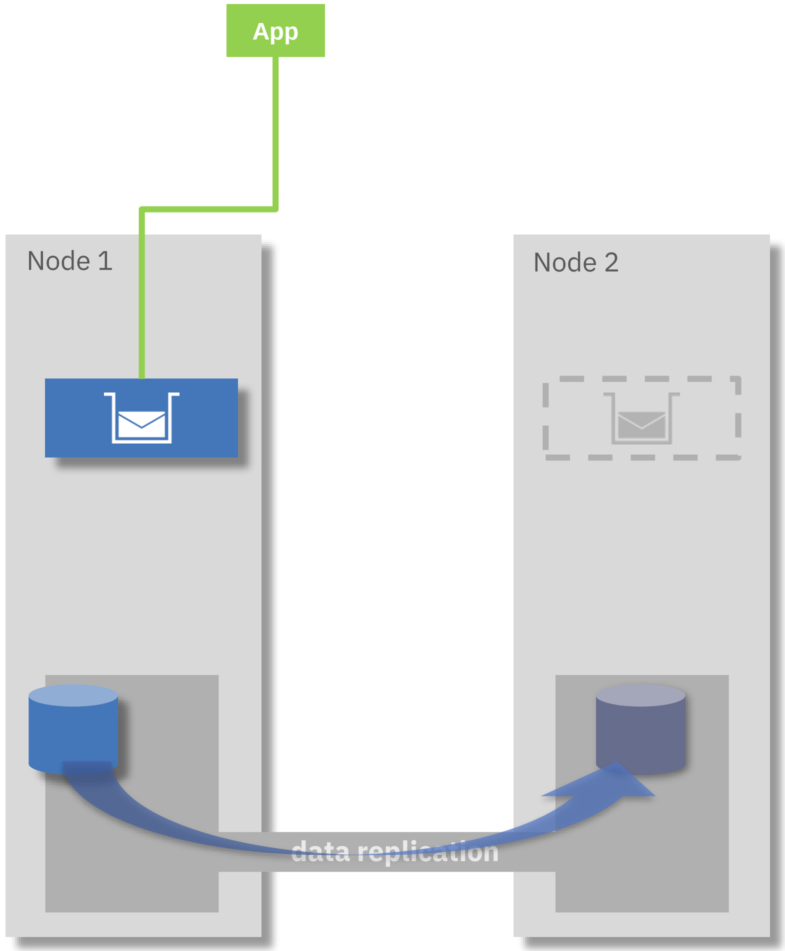
Disaster recovery

9.0.5 CD MQ Advanced added the ability to build a looser coupled pair of nodes for data replication with manual failover

Data replication can be

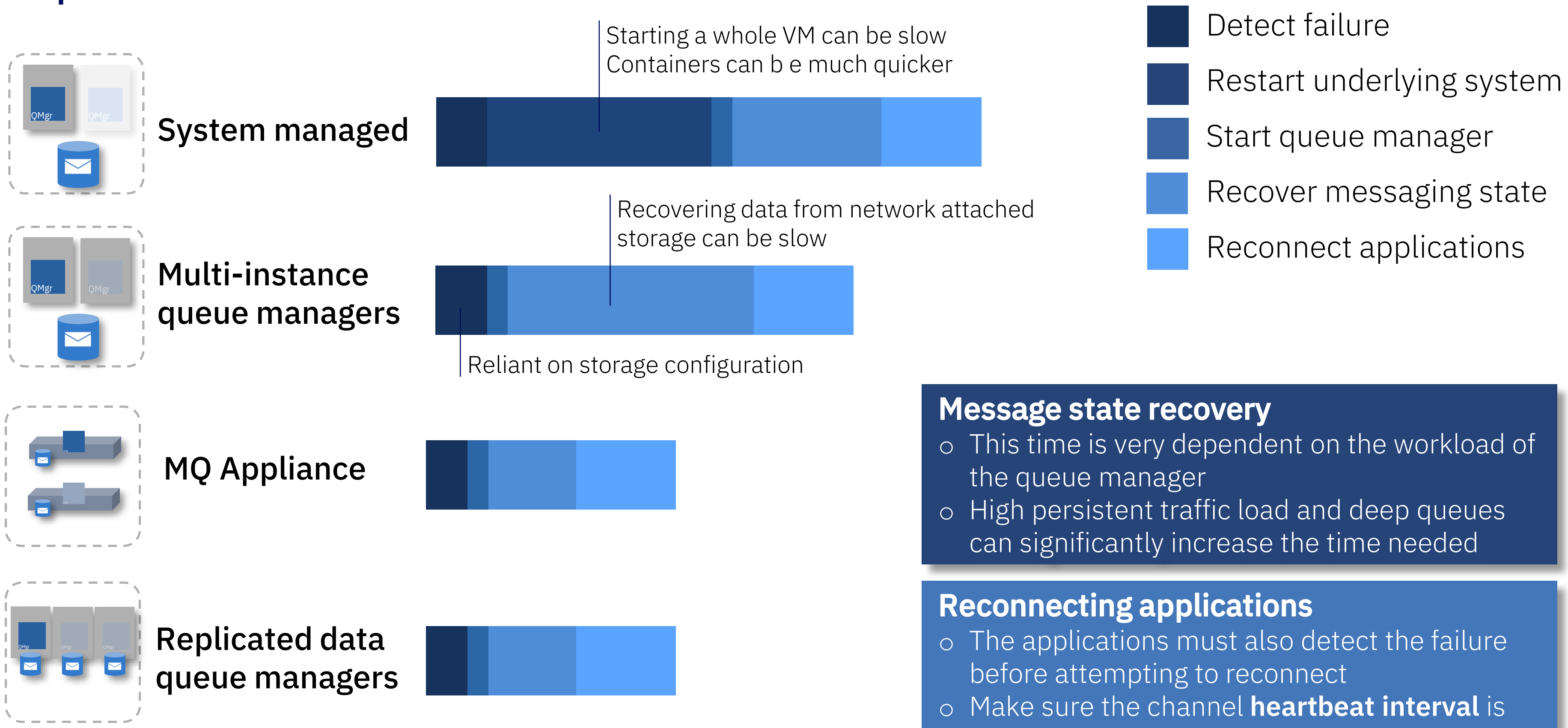
> **Asynchronous** for systems separated by a high latency network
>
> **Synchronous** for systems on a low latency network

No automatic takeover means no need for a third node to provide a quorum



Cost of a restart

Speed of failover



Message state recovery

- This time is very dependent on the workload of the queue manager
- High persistent traffic load and deep queues can significantly increase the time needed

Reconnecting applications

- The applications must also detect the failure before attempting to reconnect
- Make sure the channel **heartbeat interval** is set suitably low

Multi Instance vs. RDQM

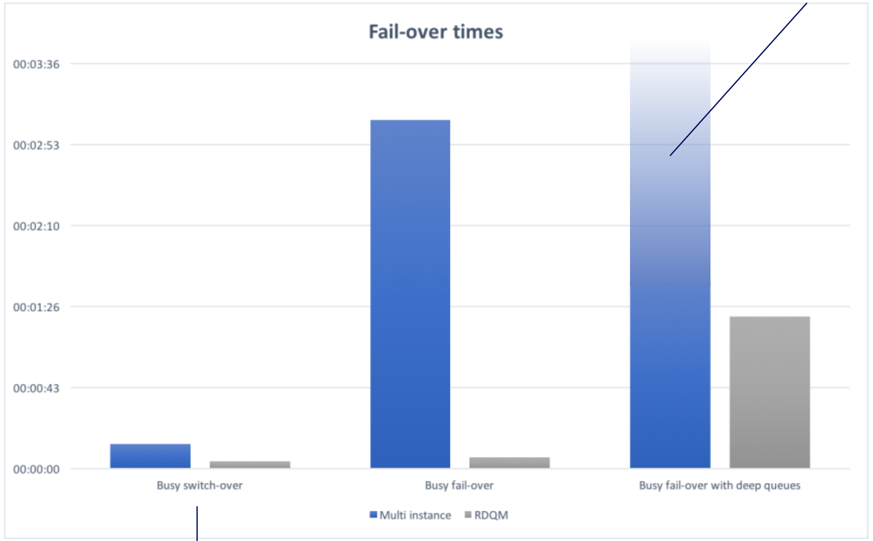
The speed of the filesystem plays a large part in the speed of failover

Two load factors effect the restart time:

- How much inflight message traffic at the time of the failure
- The amount of message data stored on active queues

The faster filesystem possible with RDQM can significantly reduce fail-over times

Misleading number due to reduction in load thanks to slower filesystem. Reality is that it'll be even slower



Even if the system is busy, a controlled switch-over gives MQ the chance to offload the inflight work

🟦 Multi instance
⬜ RDQM

What, where?

Which HA fits where

	Shared queues	System managed	Multi instance	RDQM	Appliance HA
z/OS	√				
Distributed platforms		√	√	√**	
Containers		√*	√		
MQ Appliance					√

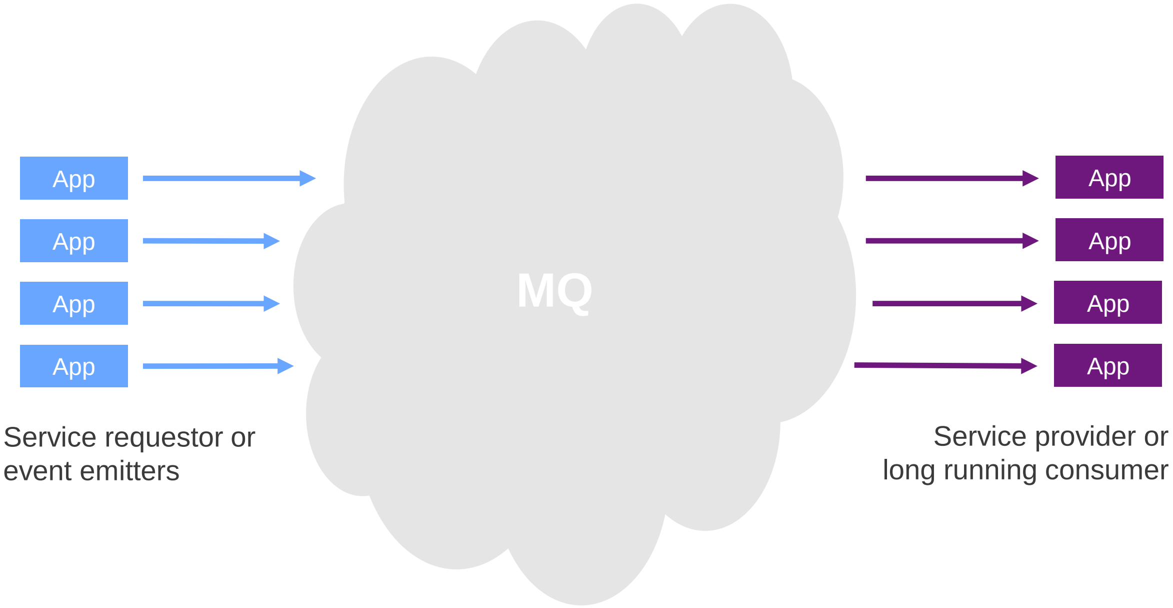
* This will depend heavily on the capabilities of the container management layer

** RHEL x86 only

Pulling it all together

Building a highly available system

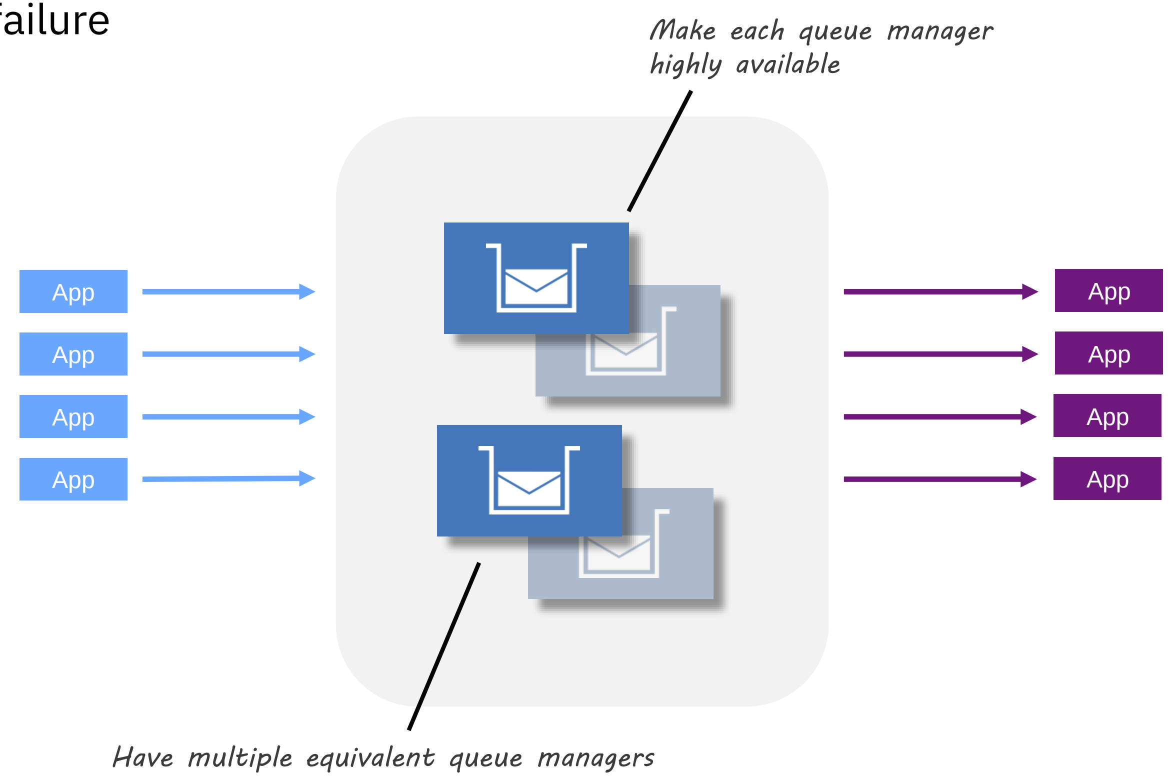
Decouple the applications from the underlying MQ infrastructure


App
App
App
App
MQ
App
App
App
App
Service requestor or event emitters
Service provider or long running consumer

Building a highly available system

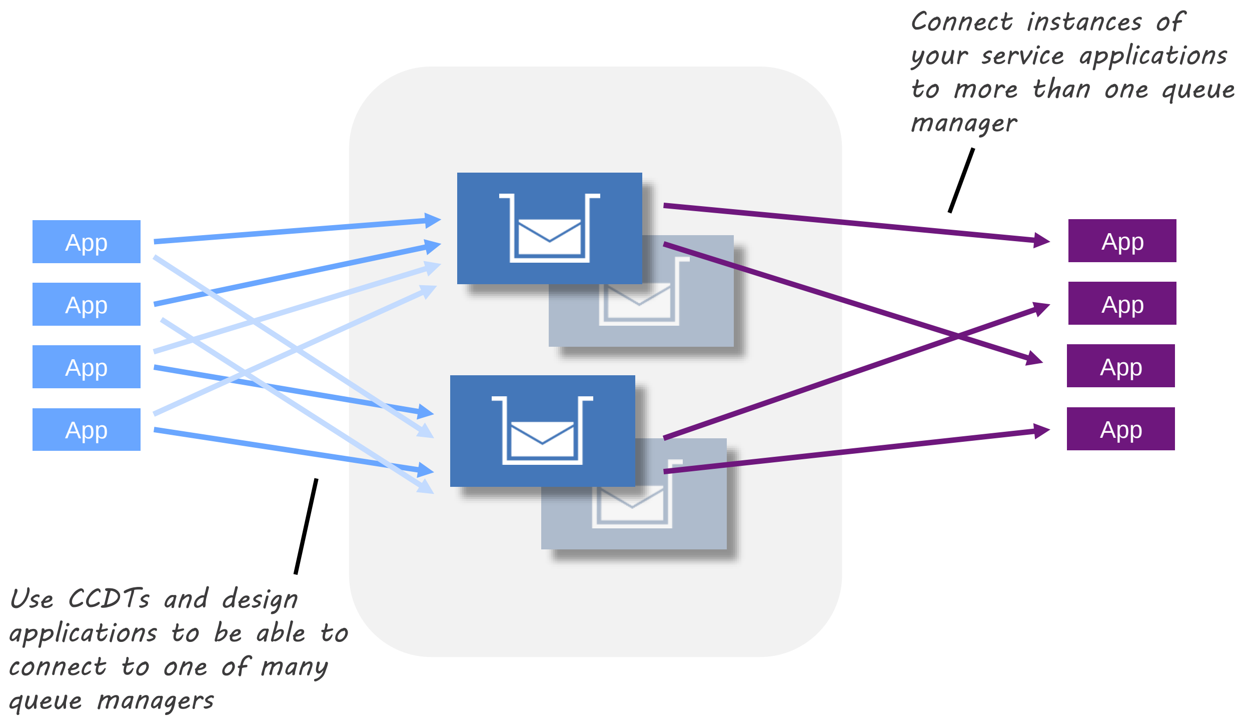
High availability of the individual MQ runtimes should be baked into the design

– Remove any single point of failure


Make each queue manager highly available
App
App
App
App
App
App
App
App
Have multiple equivalent queue managers

Building a highly available system

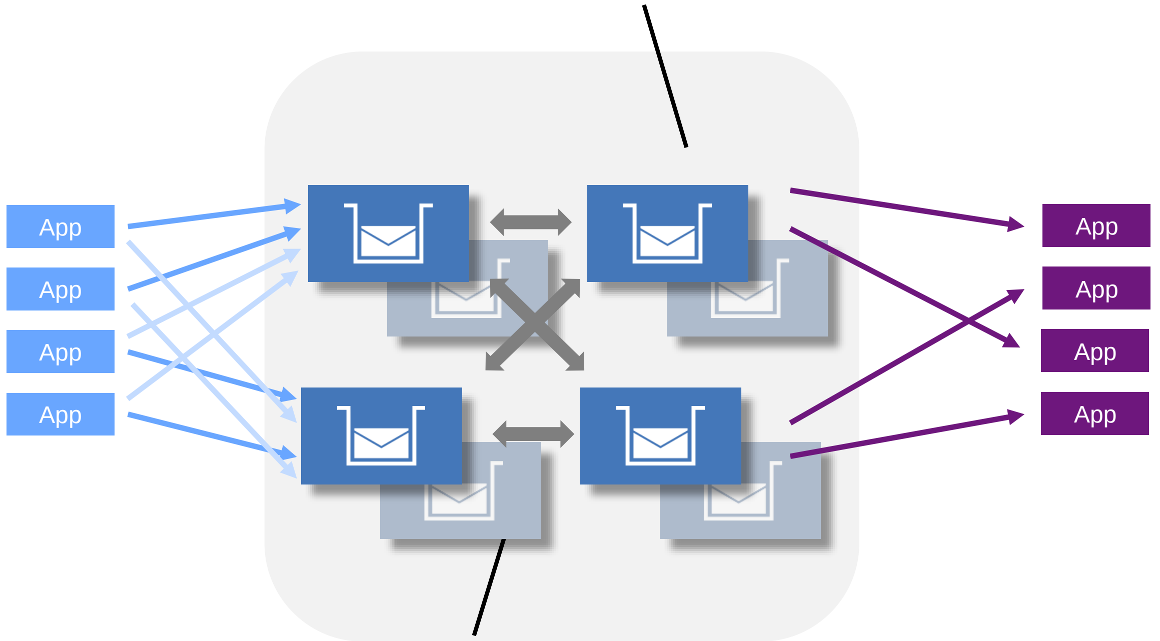
The applications should be designed and configured to maximise the availability of the MQ runtime


Connect instances of your service applications to more than one queue manager
App
App
App
App
App
App
App
App
Use CCDTs and design applications to be able to connect to one of many queue managers

Building a highly available system

Separating out the MQ infrastructure into layers can be used to improve availability further by minimising impact to end users

Separate out the requestor/emitter queue managers from the service provider queue managers



Use an MQ cluster to workload balance and dynamically route work to the service providers

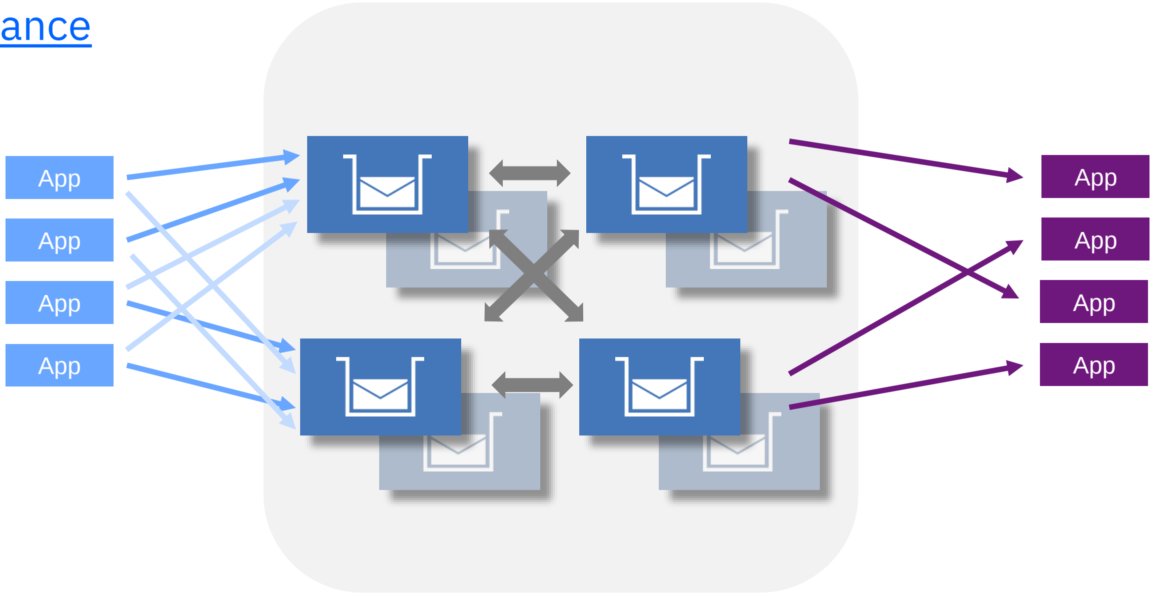
Building a highly available system

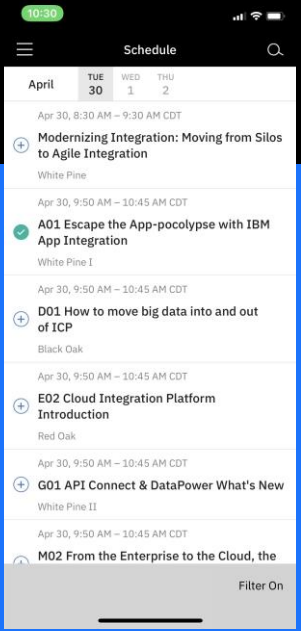
Reading material

http://ibm.biz/mq_hubs

http://ibm.biz/mqaas_red

http://ibm.biz/mqfaulttolerance




10:30

Schedule

April TUE 30 WED 1 THU 2

Apr 30, 8:30 AM – 9:30 AM CDT
Modernizing Integration: Moving from Silos to Agile Integration
White Pine

Apr 30, 9:50 AM – 10:45 AM CDT
A01 Escape the App-pocolypse with IBM App Integration
White Pine I

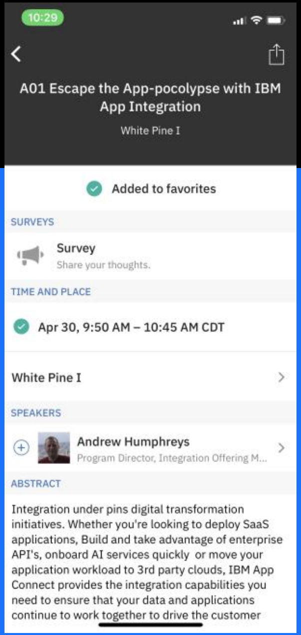
Apr 30, 9:50 AM – 10:45 AM CDT
D01 How to move big data into and out of ICP
Black Oak

Apr 30, 9:50 AM – 10:45 AM CDT
E02 Cloud Integration Platform Introduction
Red Oak

Apr 30, 9:50 AM – 10:45 AM CDT
G01 API Connect & DataPower What's New
White Pine II

Apr 30, 9:50 AM – 10:45 AM CDT
M02 From the Enterprise to the Cloud, the

Filter On


10:29

A01 Escape the App-pocolypse with IBM App Integration
White Pine I

Added to favorites

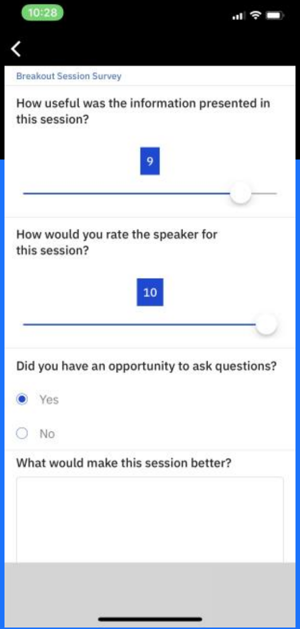
SURVEYS
Survey
Share your thoughts.

TIME AND PLACE
Apr 30, 9:50 AM – 10:45 AM CDT

White Pine I

SPEAKERS
Andrew Humphreys
Program Director, Integration Offering M...

ABSTRACT
Integration under pins digital transformation initiatives. Whether you're looking to deploy SaaS applications, Build and take advantage of enterprise API's, onboard AI services quickly or move your application workload to 3rd party clouds, IBM App Connect provides the integration capabilities you need to ensure that your data and applications continue to work together to drive the customer


10:28

Breakout Session Survey

How useful was the information presented in this session?
9

How would you rate the speaker for this session?
10

Did you have an opportunity to ask questions?
[✓ Yes] No

What would make this session better?

IBM

Don't forget to fill out the survey!

Select your session, select survey, rate the session and submit!

Thank You