

# *Deploying MQ in a self-service way*

***David Ware***

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

# Agenda

- Business drivers
- Self service of MQ resources
- Running MQ *as-a-service*

# Why?

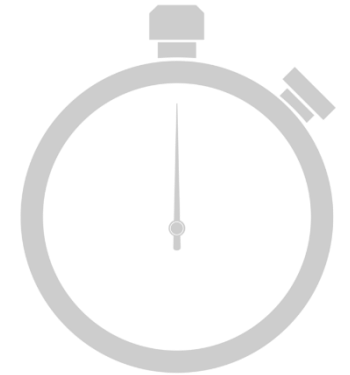
Enterprises need increased *agility* with improved administration *efficiency*

*“Applications deployed in 3 hours, not 3 months”*

*“Same size middleware team, more and more applications”*

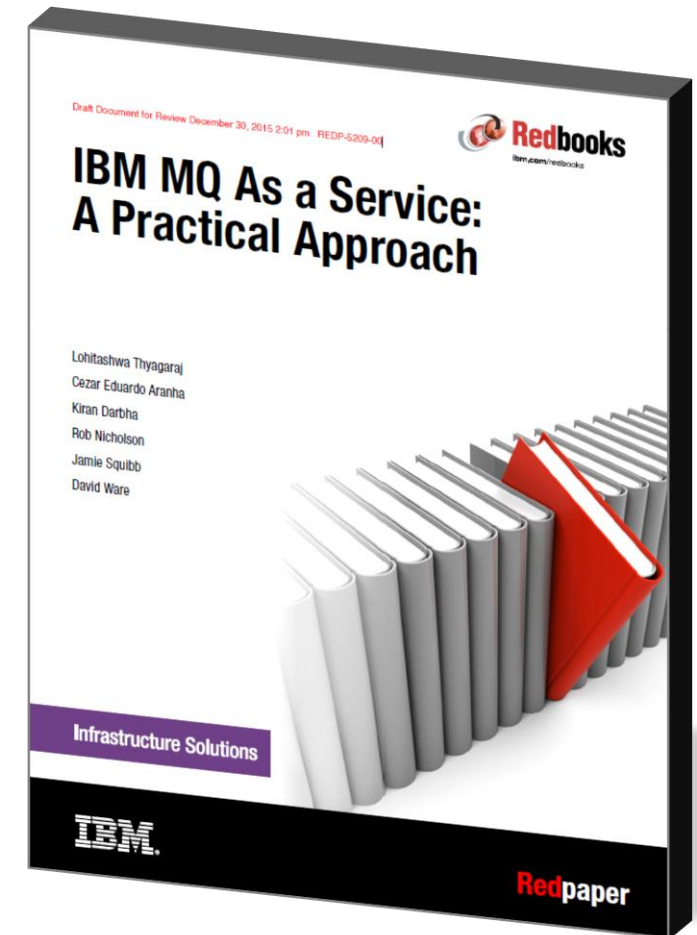
*“The system must be able to respond to change”*

- Business pressures on the infrastructure teams to do more with less
- Low MQ skills with application owners
- MQ must fit into a cloud deployment



# How?

- MQ infrastructure teams must increase the speed of delivery to their customers
- MQ complexities need hiding from the application teams
- Confidence in the MQ infrastructure must be maintained
  
- But how do you get that from MQ?
  - ▶ **Self service**
    - Providing a portal for self-service provisioning of MQ resources
  - ▶ **As a service**
    - Adopt a *service* style messaging architecture



[http://ibm.biz/mqaas\\_red](http://ibm.biz/mqaas_red)

# *A real example*

# The MQ Light Service in IBM Bluemix

- IBM runs a fully managed messaging service for users in Bluemix
- Providing applications access to asynchronous messaging using the MQ Light API
- This is underpinned by a highly available, fully automated MQ environment...



MQ Light

IBM

# Select yourself a service

IBM Bluemix

DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY

ORG: dware@u... Type to search

**Starters**

- Boilerplates

**Compute**



















- Runtimes
- Containers

**Services**

- Watson
- Mobile
- DevOps
- Web and Application
- Network
- Integration
- Data and Analytics
- Security
- Storage
- Business Analytics
- Internet of Things

**Provider**

**Web and Application**  
Deliver new web and mobile apps

 Business Rules IBM	 Data Cache IBM	 Message Hub IBM	 MQ Light IBM	 Session Cache IBM	 WebSphere Application Server IBM
 Workflow IBM	 Workload Scheduler IBM	 box Third Party	 CloudAMQP Third Party	 DreamFace Third Party	 Geocoding Third Party
 Memcached Cloud Third Party	 Moni.ai Third Party	 PubNub Third Party	 Reappt from Push Technology Third Party	 Reverse Geocoding Third Party	 Searchly Third Party

# Create an instance of the service

← Back to All Categories



## MQ Light

IBM

PUBLISH DATE  
02/05/2016

AUTHOR  
IBM

TYPE  
Service

LOCATION  
US South

[VIEW DOCS](#)

Develop responsive, scalable applications with a fully-managed messaging provider in the cloud. Quickly integrate with application frameworks through easy-to-use APIs.

- **Easy to Use**

Connect applications simply and efficiently so they can off-load work, share data or push events with a simple API and zero administration.

- **Robust and Scalable**

Rely on MQ Light's data integrity and asynchronous delivery to ensure your distributed applications are loosely-coupled, robust and scalable.

### Pick a plan

Monthly prices shown are for country or region: [United Kingdom](#)

Plan	Features
✓ MQ Light Standard Plan	Free allowance of 10,000 messages per month £3.02 GBP/Million digital messages

This is the standard service plan for MQ Light charged in units of millions of messages per month.

[TERMS](#)

### Add Service

Space:

dev

App:

Leave unbound

Service name:

MQ Light-cr

Credential name:

Credentials-1

Selected Plan:

MQ Light Standard Plan

[CREATE](#)



# Within seconds the service is there

The image shows the IBM Bluemix dashboard interface. At the top, there is a navigation bar with links for DASHBOARD, SOLUTIONS, CATALOG, PRICING, DOCS, and COMMUNITY. The user's organization is identified as 'dware@uk.ibm.com'. The main dashboard area features three large blue buttons: 'CREATE APP', 'START CONTAINERS', and 'RUN VIRTUAL MACHINES'. Below these are two dark grey cards: 'Data & Analytics' with a 'WORK WITH DATA' button, and 'Services & APIs' with a '100+' badge, '1/4 Used' status, and a 'USE SERVICES OR APIS' button. A left sidebar lists resource counts: 'dev' space, 'CF APPS (0)', 'SERVICES (1)', 'CONTAINERS (0)', and 'VIRTUAL MACHINES (0)'. At the bottom, a 'Services' section is visible, with one service card highlighted by a red border. This card is for 'MQ Light-pj', an 'MQ Light' service, currently in an 'Unbound Service' state, with a 'standard' plan and a star icon.

IBM Bluemix

DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY

ORG: dware@uk.ibm.com

+ Create a Space

dev

- CF APPS (0)
- SERVICES (1)
- CONTAINERS (0)
- VIRTUAL MACHINES (0)

CREATE APP START CONTAINERS RUN VIRTUAL MACHINES

Data & Analytics

WORK WITH DATA

Services & APIs

100+ 1/4 Used

USE SERVICES OR APIS

Services

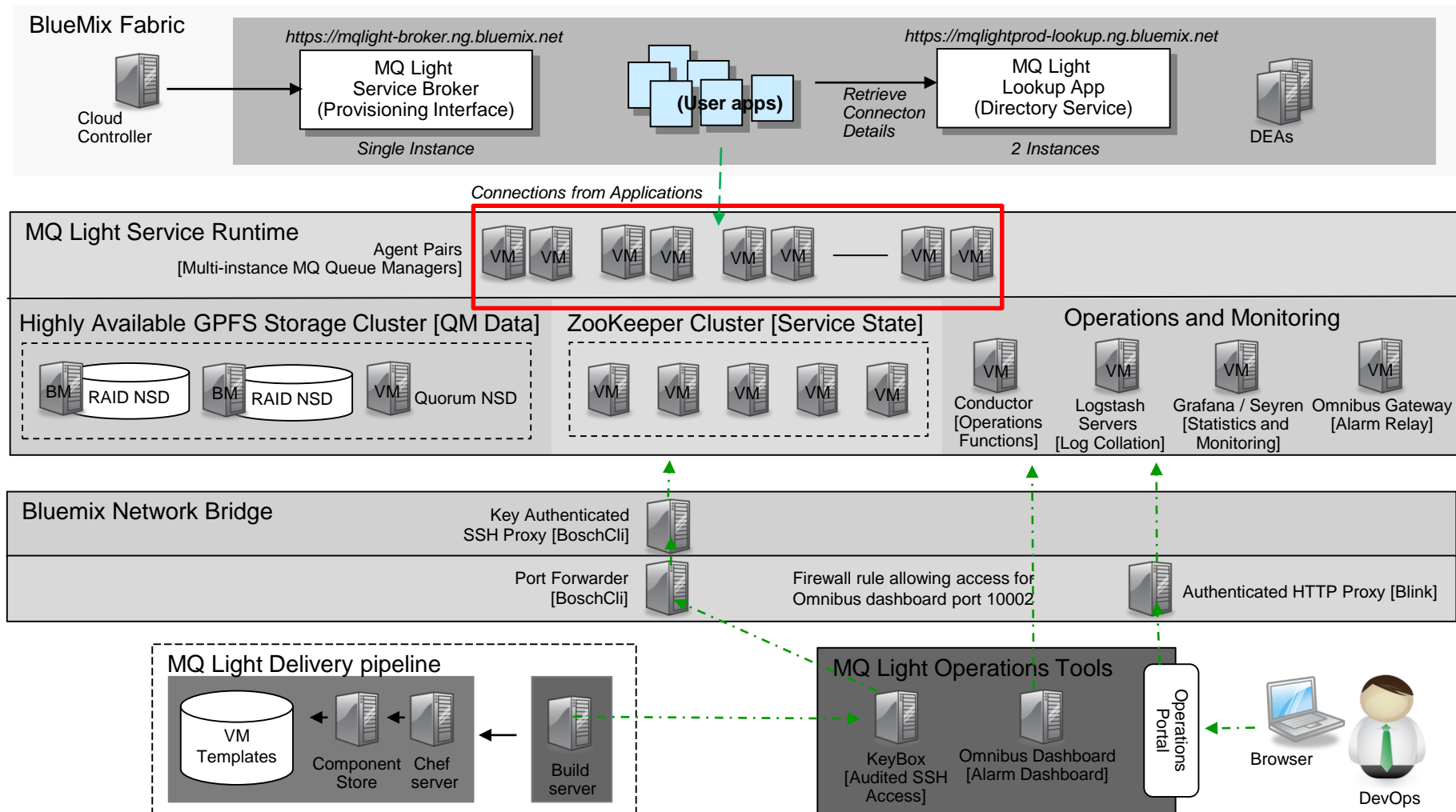
MQ Light-pj

MQ Light

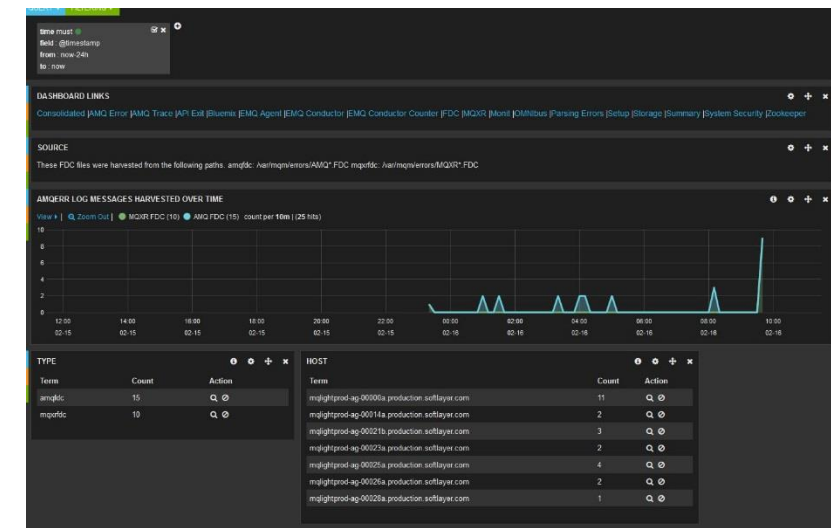
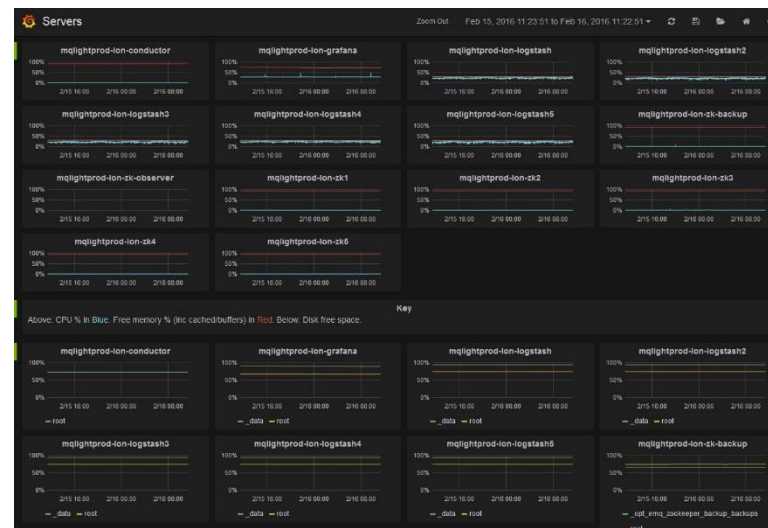
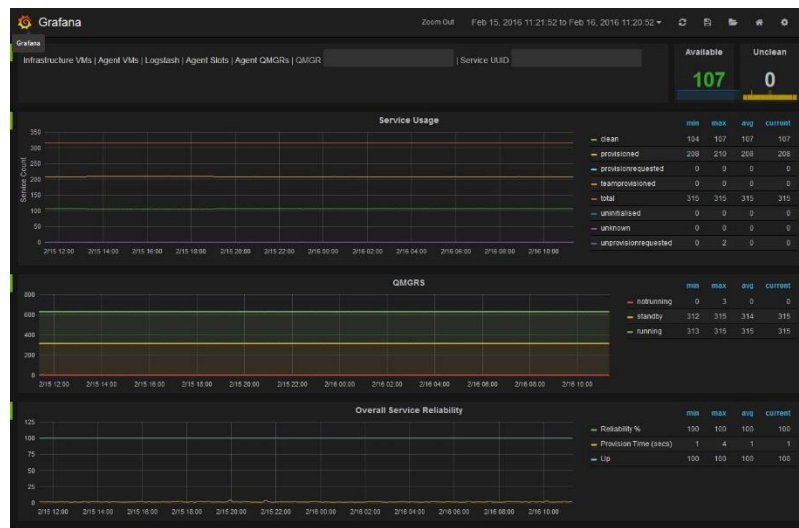
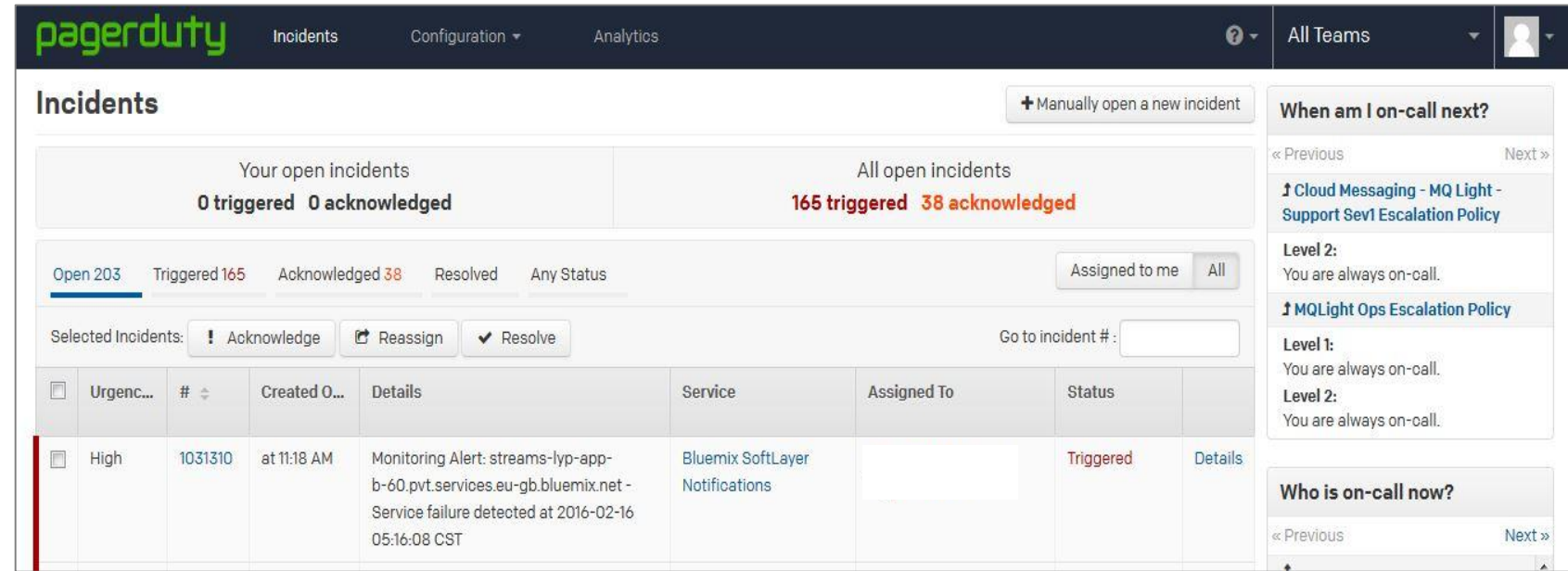
Unbound Service

Plan: standard

# But behind the scenes



# Which automatically ties directly into our operations



# *Self service*

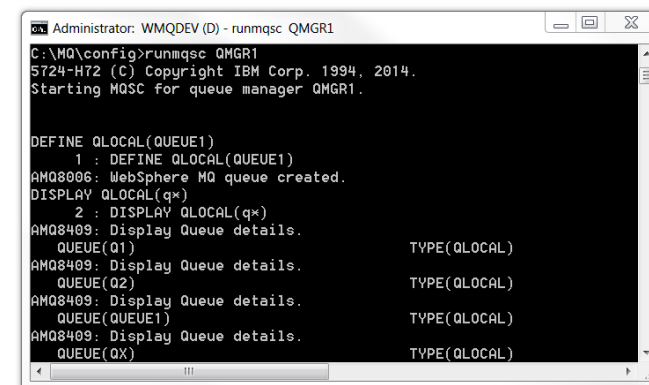
# Self service

- Who is this 'self'?
  - ▶ Typically the application owning teams
  - ▶ But could be the administration team
- Scope of self service?
  - ▶ Development environments
  - ▶ Test systems
  - ▶ Production environments
- Extent of what to self serve?
  - ▶ Simple provisioning of queues?
  - ▶ Provisioning of whole queue managers and architectures?



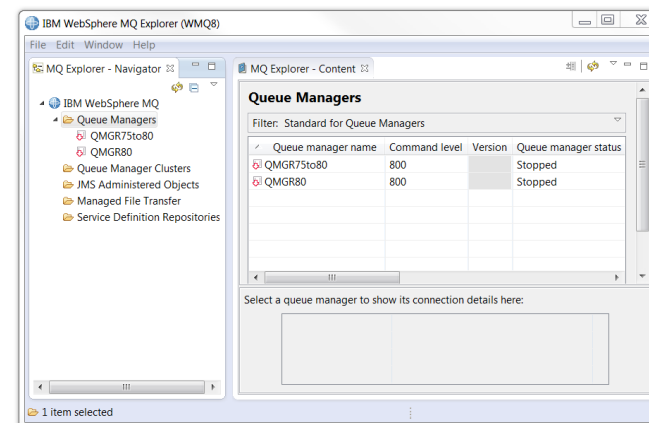
# Self service tooling

- *From one point of view MQ's "self service portal" is simply its traditional administrative tooling*
  - ▶ I.e. *MQ Explorer, runmqsc*
  - ▶ This provides remote access to MQ configuration, with per-user/group authorisation for visibility and modification.
- Some may expose all this to their development users, *but you really shouldn't...*
- Is that something you really want to do?
  - ▶ You can't expect your users to know all there is to know about MQ configuration
  - ▶ You probably only want to expose a subset of capabilities
  - ▶ MQ is not the centre of the universe! You'll need to coordinate MQ resources with the other components in the solution anyway



```
Administrator: WMQDEV (D) - runmqsc QMGR1
C:\MQ\config>runmqsc QMGR1
5724-HT2 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager QMGR1.

DEFINE QLOCAL(Queue1)
  1 : DEFINE QLOCAL(Queue1)
AMQ8006: WebSphere MQ queue created.
DISPLAY QLOCAL(q*)
  2 : DISPLAY QLOCAL(q*)
AMQ8409: Display Queue details.
        QUEUE(Q1)                                TYPE(QLOCAL)
AMQ8409: Display Queue details.
        QUEUE(Q2)                                TYPE(QLOCAL)
AMQ8409: Display Queue details.
        QUEUE(Queue1)                            TYPE(QLOCAL)
AMQ8409: Display Queue details.
        QUEUE(QX)                                TYPE(QLOCAL)
```



# Self service tooling

- So from that point of view, MQ doesn't have self service tooling
- And as no off the shelf solution will 100% match your corporation's requirements you'll need to build/customise your own
  - ▶ And **many** have built such tooling over the years
- Options for coordinated self service provisioning
  - UrbanCode Deploy
  - z/OSMF
  - Pure Application Systems
  - Chef, Puppet, ...
  - Third party tooling
  - Hand rolled
- Instead, MQ provides the interfaces to make using these possible...



# MQ's administrative interfaces

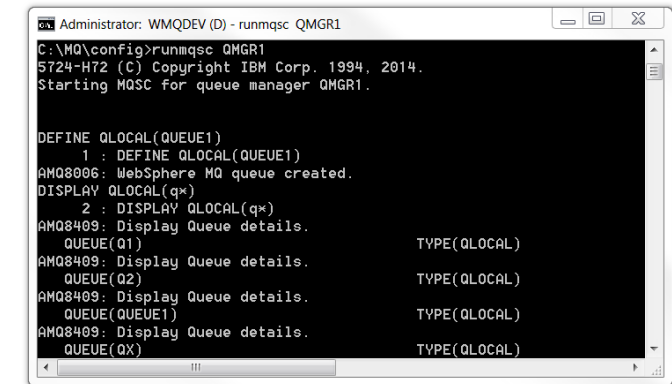
- MQ provides comprehensive interfaces for automating administrative changes and monitoring

- Scripting

- ▶ MQSC
- ▶ Pipe commands into **runmqsc**
- ▶ Runs locally or remotely [v8]

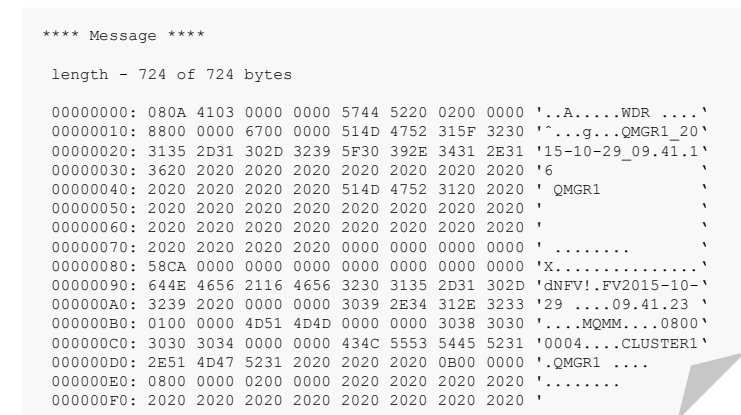
- Programmatically

- ▶ Programmable Command Format (**PCF**)
- ▶ Use messaging to send administrative commands to queue managers
- ▶ Consume messages containing monitoring, statistics and event information



```
Administrator: WMQDEV (D) - runmqsc QMGR1
C:\MQ\config>runmqsc QMGR1
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager QMGR1.

DEFINE QLOCAL(Queue1)
  1 : DEFINE QLOCAL(Queue1)
AMQ8006: WebSphere MQ queue created.
DISPLAY QLOCAL(q*)
  2 : DISPLAY QLOCAL(q*)
AMQ8409: Display Queue details.
        QUEUE(Q1)                                TYPE(QLOCAL)
AMQ8409: Display Queue details.
        QUEUE(Q2)                                TYPE(QLOCAL)
AMQ8409: Display Queue details.
        QUEUE(Queue1)                            TYPE(QLOCAL)
AMQ8409: Display Queue details.
        QUEUE(QX)                                TYPE(QLOCAL)
```



```
**** Message ****

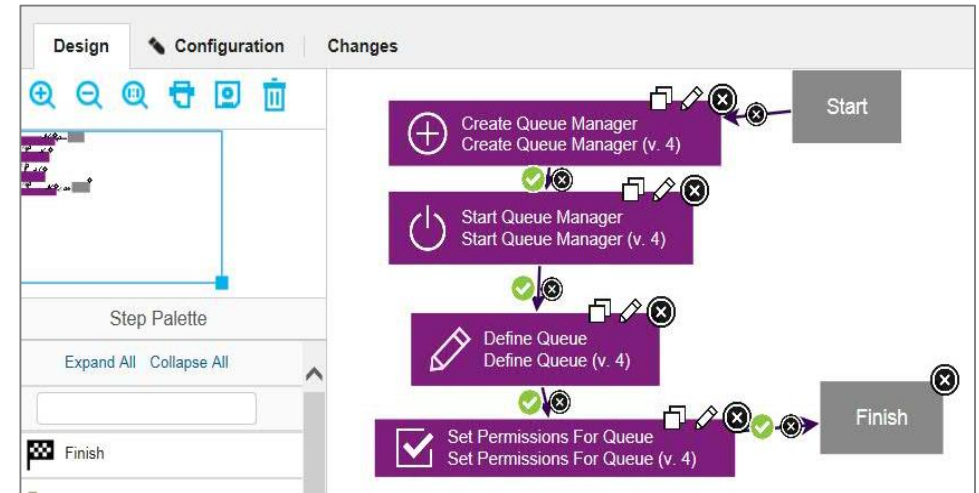
length - 724 of 724 bytes

00000000: 080A 4103 0000 0000 5744 5220 0200 0000 '..A....WDR ....'
00000010: 8800 0000 6700 0000 514D 4752 315F 3230 '^...g...QMGR1_20'
00000020: 3135 2D31 302D 3239 5F30 392E 3431 2E31 '15-10-29_09.41.1'
00000030: 3620 2020 2020 2020 2020 2020 2020 2020 '6'
00000040: 2020 2020 2020 2020 514D 4752 3120 2020 ' QMGR1'
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
00000060: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
00000070: 2020 2020 2020 2020 0000 0000 0000 0000 '.....'
00000080: 58CA 0000 0000 0000 0000 0000 0000 0000 'X.....'
00000090: 644E 4656 2116 4656 3230 3135 2D31 302D 'dNFV!.FV2015-10-'
000000A0: 3239 2020 0000 0000 3039 2E34 312E 3233 '29 ...09.41.23 '
000000B0: 0100 0000 4D51 4D4D 0000 0000 3038 3030 '...MQMM...0800'
000000C0: 3030 3034 0000 0000 434C 5553 5445 5231 '0004...CLUSTER1'
000000D0: 2E51 4D47 5231 2020 2020 2020 0B00 0000 '.QMGR1 ....'
000000E0: 0800 0000 0200 0000 2020 2020 2020 2020 '.....'
000000F0: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
```



# IBM UrbanCode Deploy

- UrbanCode orchestrates and automates the deployment of applications and middleware configurations into development, test and production environments.
- Coordinates deployments across multiple machines.
- MQ plugin for UrbanCode provides automations for off the peg or customised deployments of UrbanCode artefacts
  - Queue manager creation
  - Queues, channels, topics, etc.
  - Custom mqsc scripts
  - ...



The screenshot shows the 'Component Process Properties' configuration editor. A red arrow points from the 'Component Process Properties' tab in the left sidebar to the table. The table has the following data:

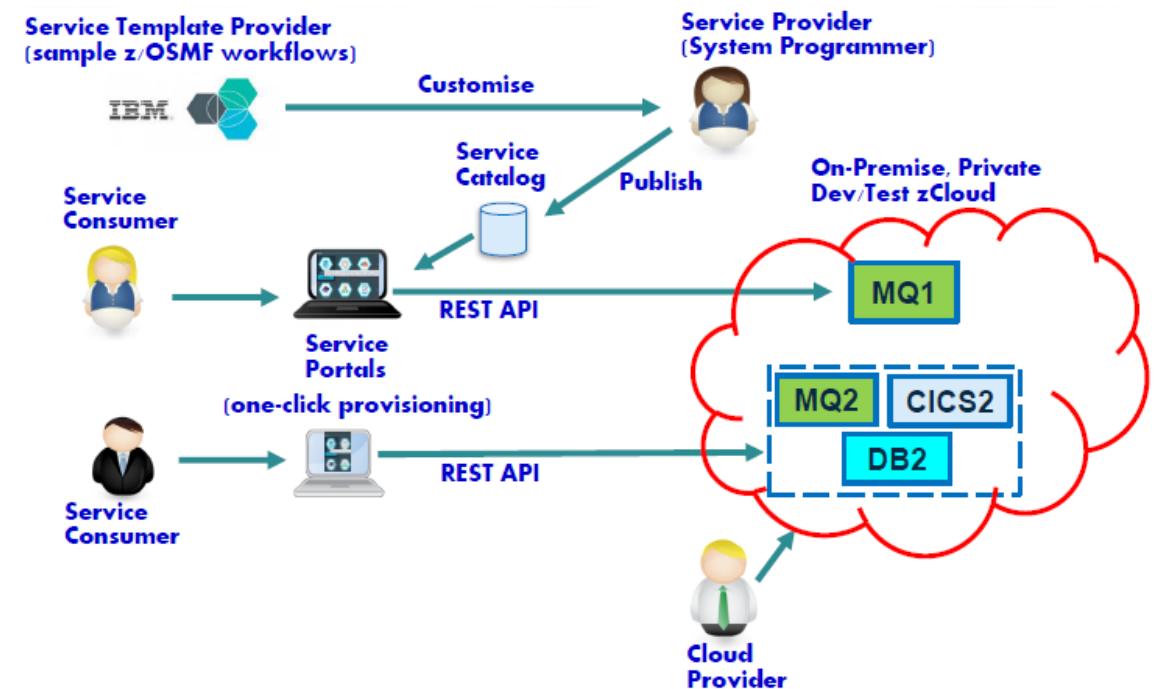
Name	Label	Pattern	Required	Default Value	Description	Actions
Port	Port		true	3000		Edit Delete
QueueName	QueueName		true			Edit Delete

At the bottom of the table, it says '2 records - Refresh Print' and 'Rows 10'.

See the MQ as-a-service redpaper for details

# Sample z/OSMF Workflows

- z/OSMF provides services to help customers rapidly provision/de-provision z/OS middleware
  - ▶ Including MQ, DB2, CICS, IMS, WAS
  - ▶ Workflows can be implemented to automate tasks
  - ▶ Self-service/click of a button
  - ▶ Rapidly stand-up/down MQ resources for development/test purposes



- Experimental IBM MQ cookbook
- Demonstrates idempotent installation on MQ installation on Linux
- Includes queue manager creation and start
  
- Feedback very welcome

<https://github.com/ibm-messaging/mq-chef>



# Self-service portal considerations

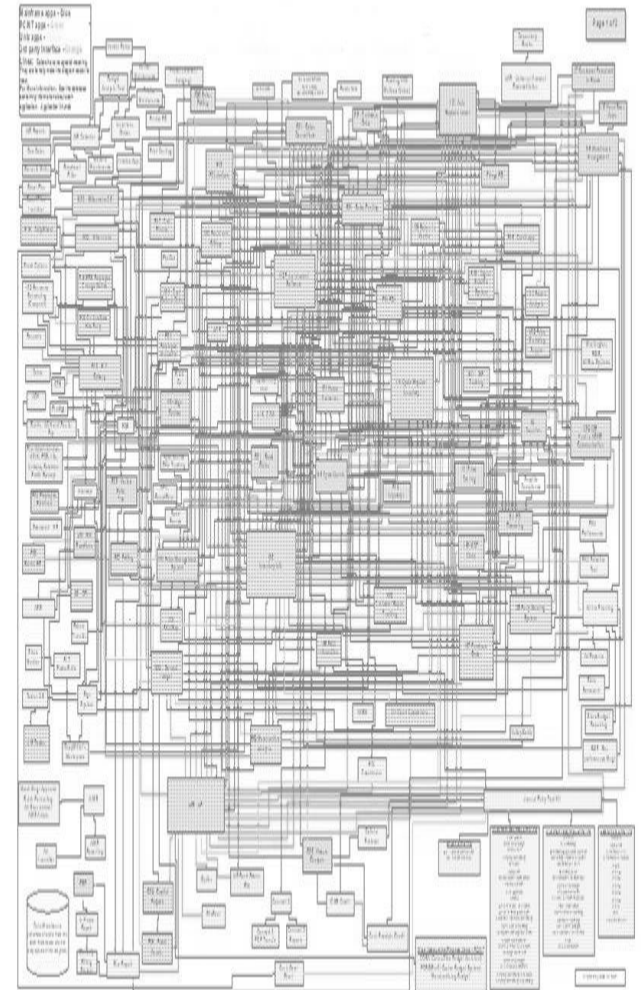
- Limit the MQ options available to users as much as possible
  - ▶ This will depend on the expected MQ skill level of *users*
  - ▶ Minimise MQ resources and attributes exposed to the users
  - ▶ Ideally abstract the underlying MQ resources from the users
  - ▶ *We'll discuss this area more later*
- Use a portal to act as an intermediary
  - ▶ Do not grant your users MQ administrator rights, *the portal is the MQ administrator*
- Log all changes
  - ▶ Either using the portal
  - ▶ Or (and) with MQ configuration events
- Integrate resources direct into the monitoring system
  - ▶ When provisioning systems, enable background monitoring as standard

***As a service***

# As a service MQ

- MQ has near infinite variations in architectures and configurations
  - ▶ **Excellent** for allowing customisation to your company's exact requirements
  - ▶ **Bad** to expose to the end users through self-service
- To support self-service, limit those variations and customisations to a minimal set of *patterns*
  - ▶ Once the patterns have been determined, the self-service tooling should map any requests onto the appropriate pattern
  - ▶ This reduces the level of end user knowledge required to make changes
  - ▶ It also simplifies the job of the operations teams in supporting such systems

*You may need to compromise on some of your customisations and optimisations*



# MQ as a service best practices

- There is no single MQ pattern but there are good practices that can make such deployments more successful
  - ▶ Naming conventions
  - ▶ Separate the applications from the queue managers
  - ▶ Application isolation
  - ▶ MQ Clustering
  - ▶ Messaging hubs



# Naming conventions

- Pick a naming convention and enforce it
  - ▶ Use your self service portal to generate the names, don't leave it to the user
  - ▶ Helps the operations teams to understand the system even if it has been created by the application teams
  - ▶ Be mindful of the MQ 48 character, special character, restrictions
  - ▶ Map names of MQ resources into a directory to record additional information
- For example, for each resource:
  - Who is responsible for this resource?
  - What applications are dependant of this resource?
  - What role is this resource playing?
  - Unique within the application
- E.g. “**ORGA**.**APPZ**.REQUEST.Q1”

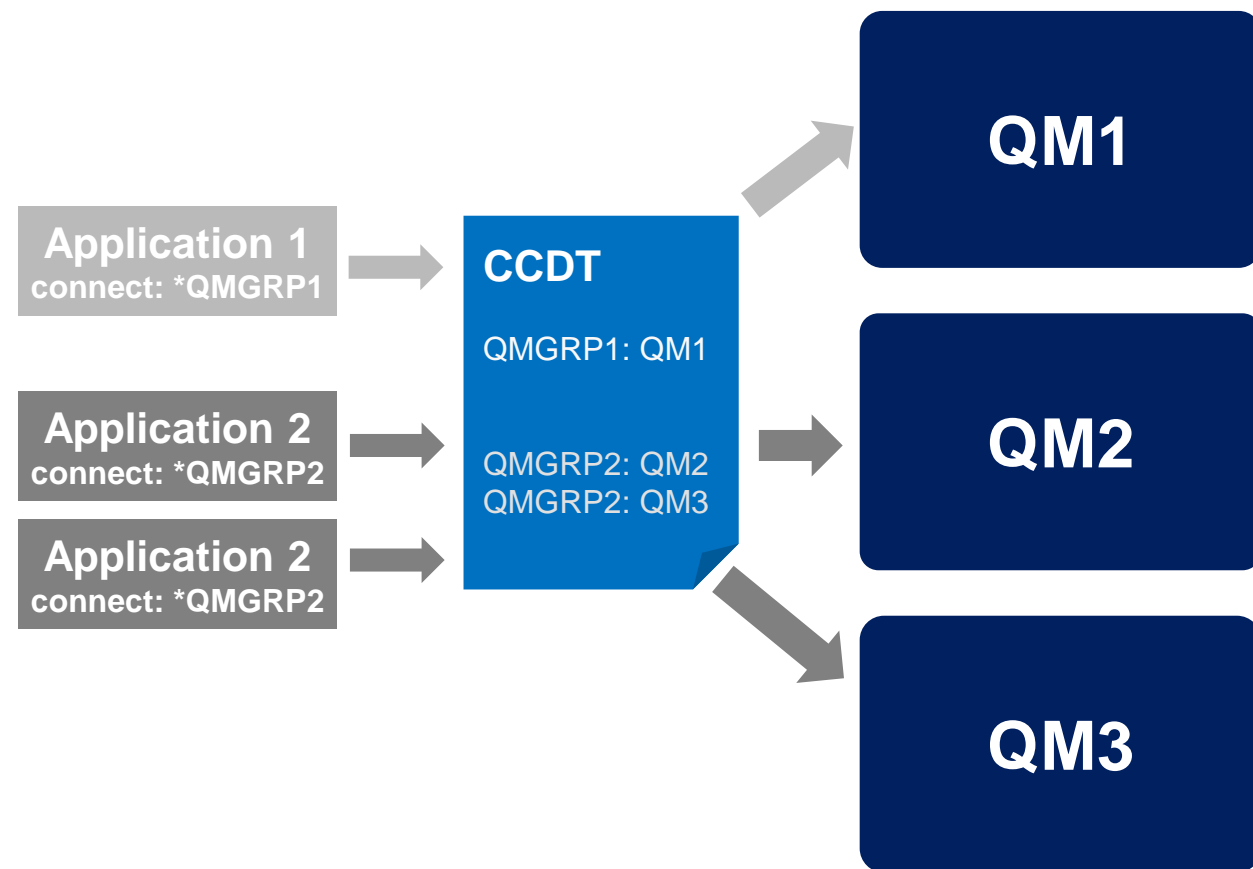


# Detach applications from queue managers

- Binding an application to a specific queue manager can restrict future changes
  - ▶ Changes to architecture
  - ▶ Changes to scale of messaging or application runtimes
  - ▶ Changes to software versions
  - ▶ Maintenance windows
- Run applications remote from the queue managers
  - ▶ Connect as clients to queue managers
- Hide specific queue manager names from the applications
  - ▶ An application should definitely not need to hard code a queue manager's name
- Use Client Channel Definition Tables (**CCDTs**)
  - ▶ These provide encapsulation of connection information
  - ▶ Centrally administered and made available to applications
  - ▶ Enable security, high availability and workload balancing of clients

# What do CCDTs enable?

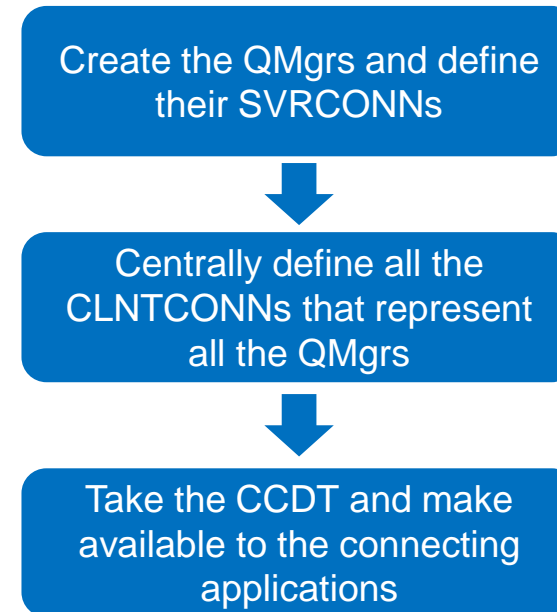
- Applications simply connect to a “queue manager” name
- MQ architecture detail hidden from the application
- CCDT defines which real queue managers the application will connect to
  - ▶ Can be a single queue manager
  - ▶ Or a group, defined in the CCDT
    - Selection can be ordered or randomised and weighted



*Use “\*” names to decouple the application from the real queue manager name*

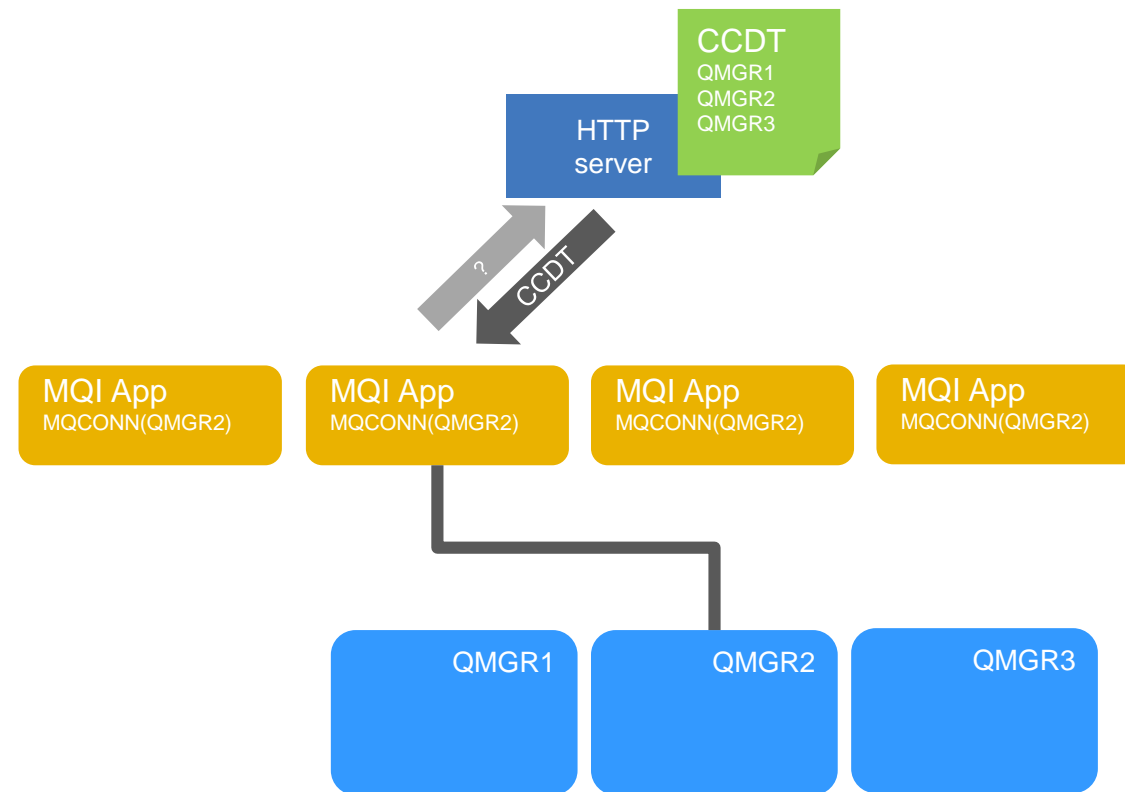
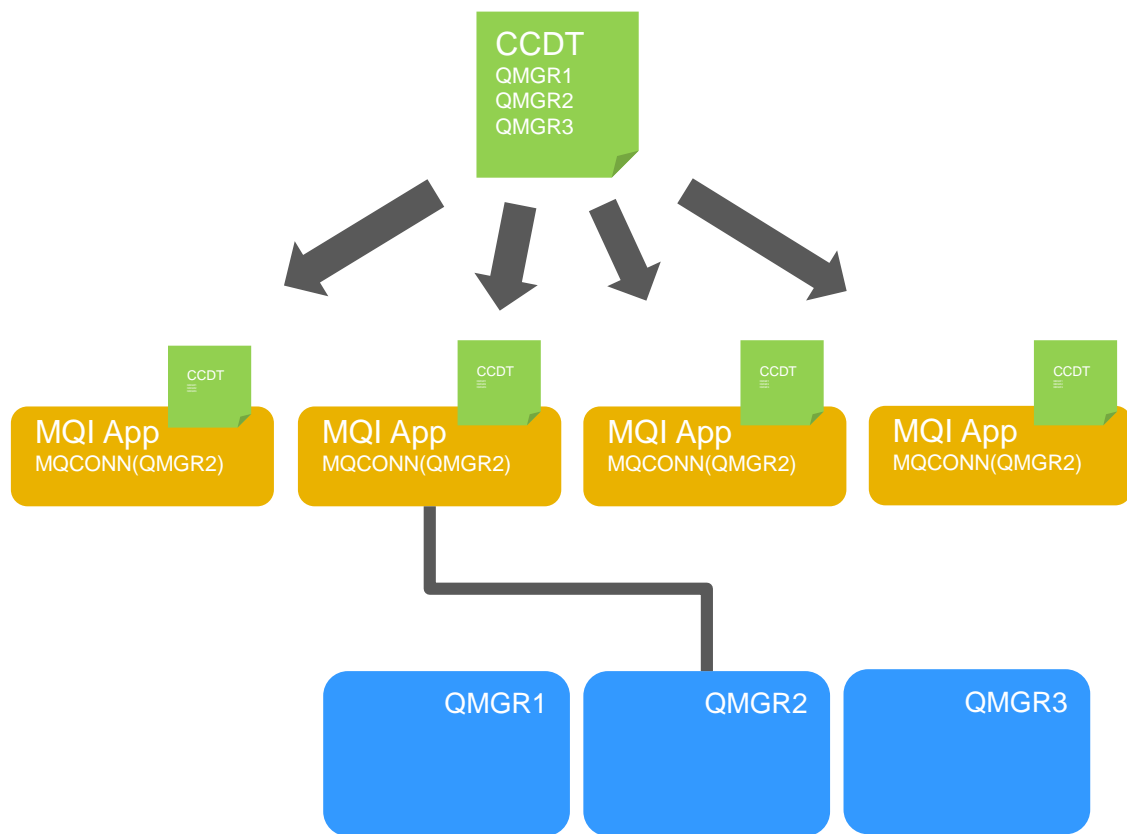
# Creating the CCDTs

- 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:** Use 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



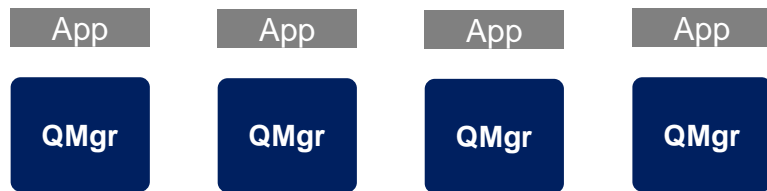
# Accessing the CCDTs

- 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**



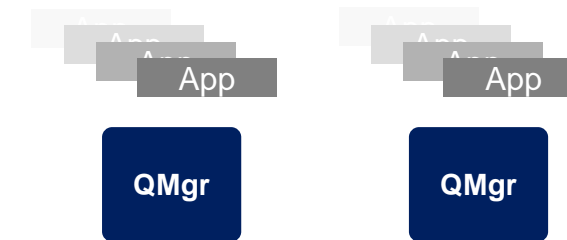
# Queue manager tenancy

- Multiple applications need isolation from each other
  - ▶ Security and impact
- Do you share queue managers or provide dedicated ones for your users?



- Single tenant

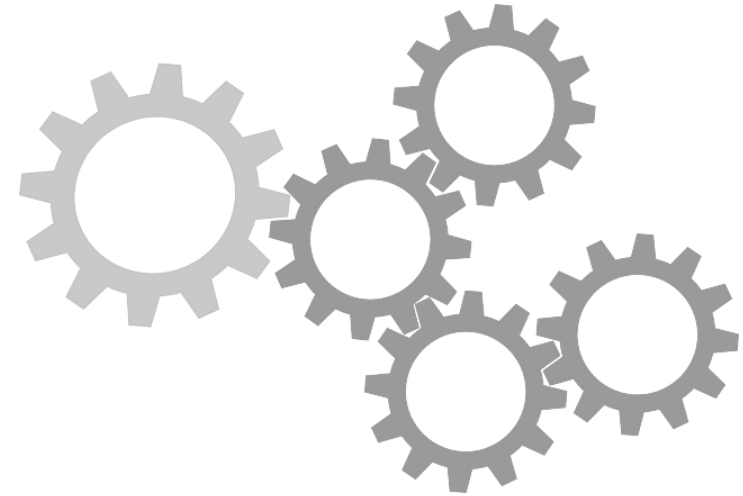
- Simple to configure, maintain and monitor
- Very good isolation
- A proliferation of queue managers
- Harder when integration is required



- Multi tenant

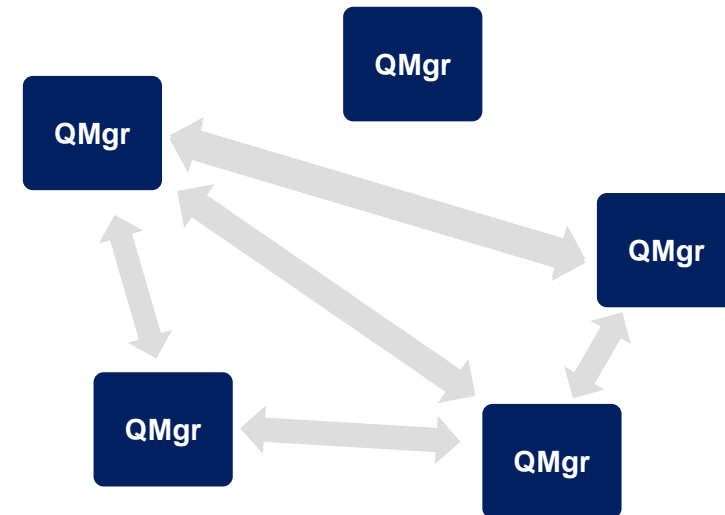
- Lower machine overheads
- More care needed in configuring to achieve isolation
  - Isolation of machine resources not possible
- Harder/simpler to monitor
  - Depends on your view of more queue managers
- Fine grain security required

- Standardize your queue managers
  - ▶ Minimise bespoke configurations
  - ▶ Groups of queue managers have roles, providing equivalent capabilities
  - ▶ Consider a layered architecture
    - For example:
      - **Connectors:** Queue managers that inbound applications connect to
      - **Service providers:** Queue managers that host the resources being serviced



# MQ Clustering for integration

- MQ clusters is the *right* way to connect queue managers together for as-a-service
- Fits the model of easily adding and removing queue managers
- Dynamically handles end-to-end routing
- Provides workload balancing for horizontal scaling and dynamic routing for continuous availability
- Drastically reduces the amount of configuration required when new resources are defined, simplifying any self service solution



# An MQ Hub topology

- Pulling all this together into the MQ “Hub” topology
  - ▶ The hub decouples the applications from the underlying MQ infrastructure

<http://ibm.biz/MQHubPart1>

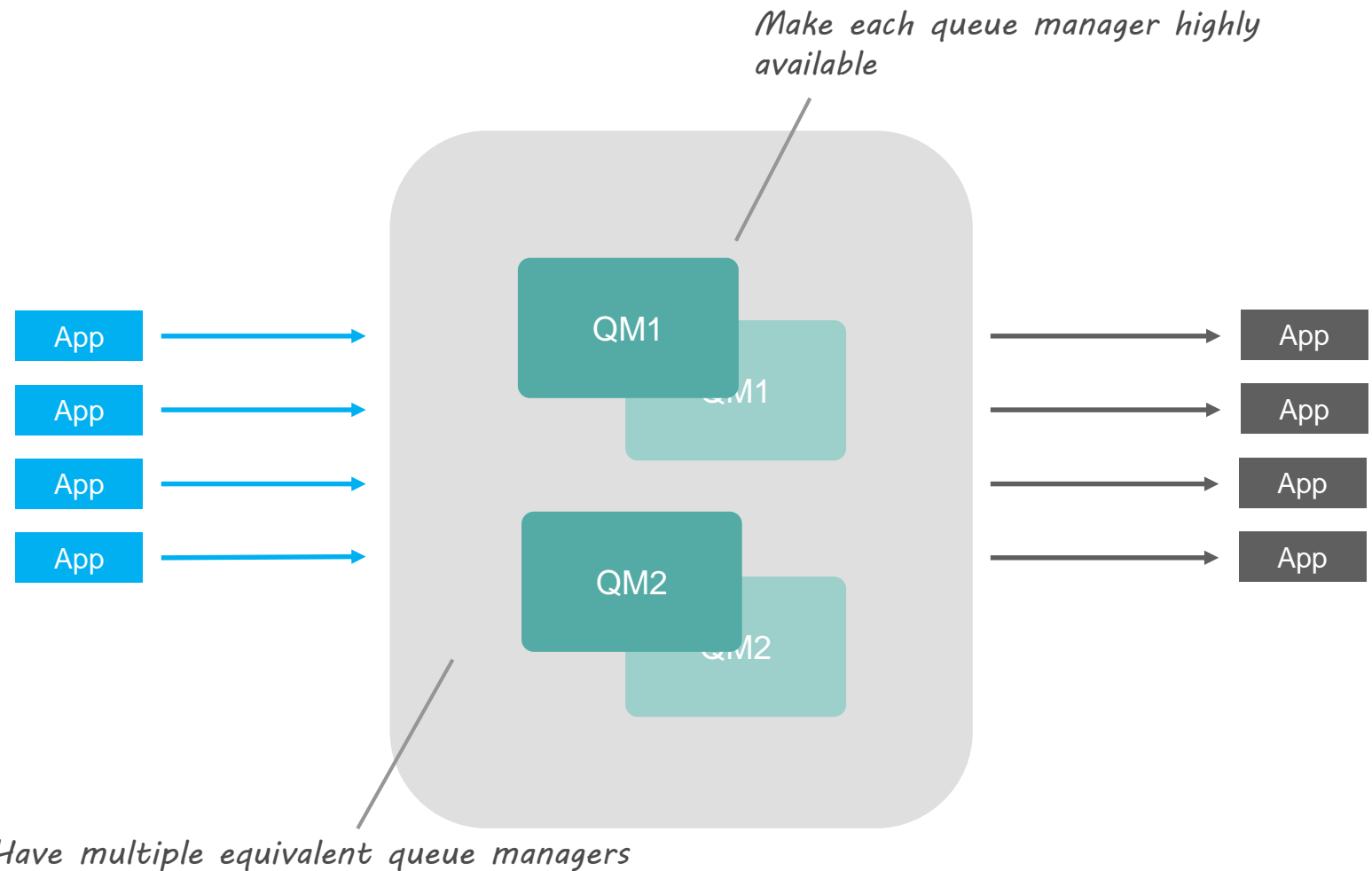




# An MQ Hub topology

- Pulling all this together into the MQ “Hub” topology
  - ▶ High availability of the MQ runtime should be baked into the hub design

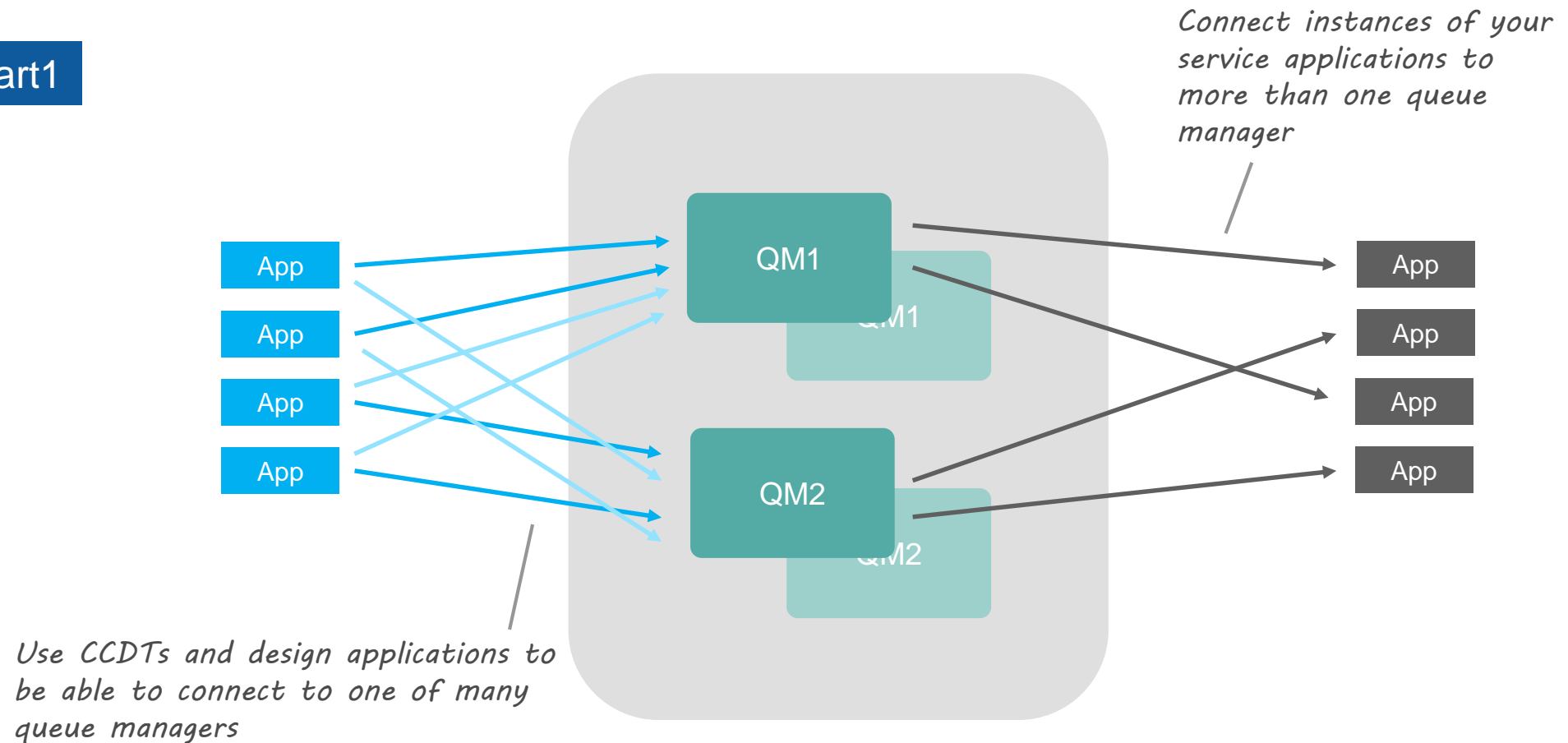
<http://ibm.biz/MQHubPart1>



# An MQ Hub topology

- Pulling all this together into the MQ “Hub” topology
  - ▶ The applications must be configured to exploit the availability of the MQ runtime

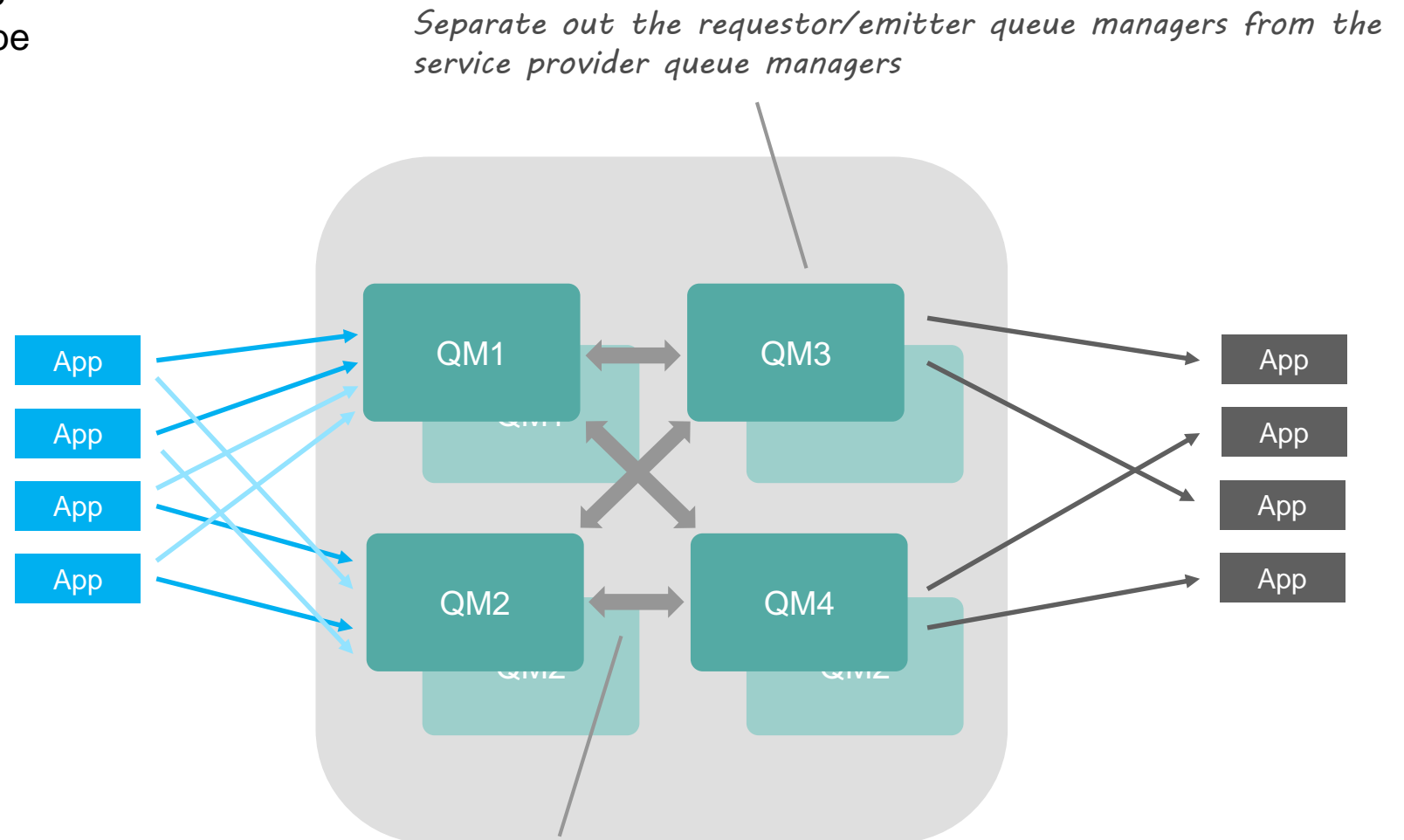
<http://ibm.biz/MQHubPart1>



# An MQ Hub topology

- Pulling all this together into the MQ “Hub” topology
  - ▶ Separating out the MQ infrastructure into front facing and service facing tiers can be used to improve workload balancing.

<http://ibm.biz/MQHubPart1>

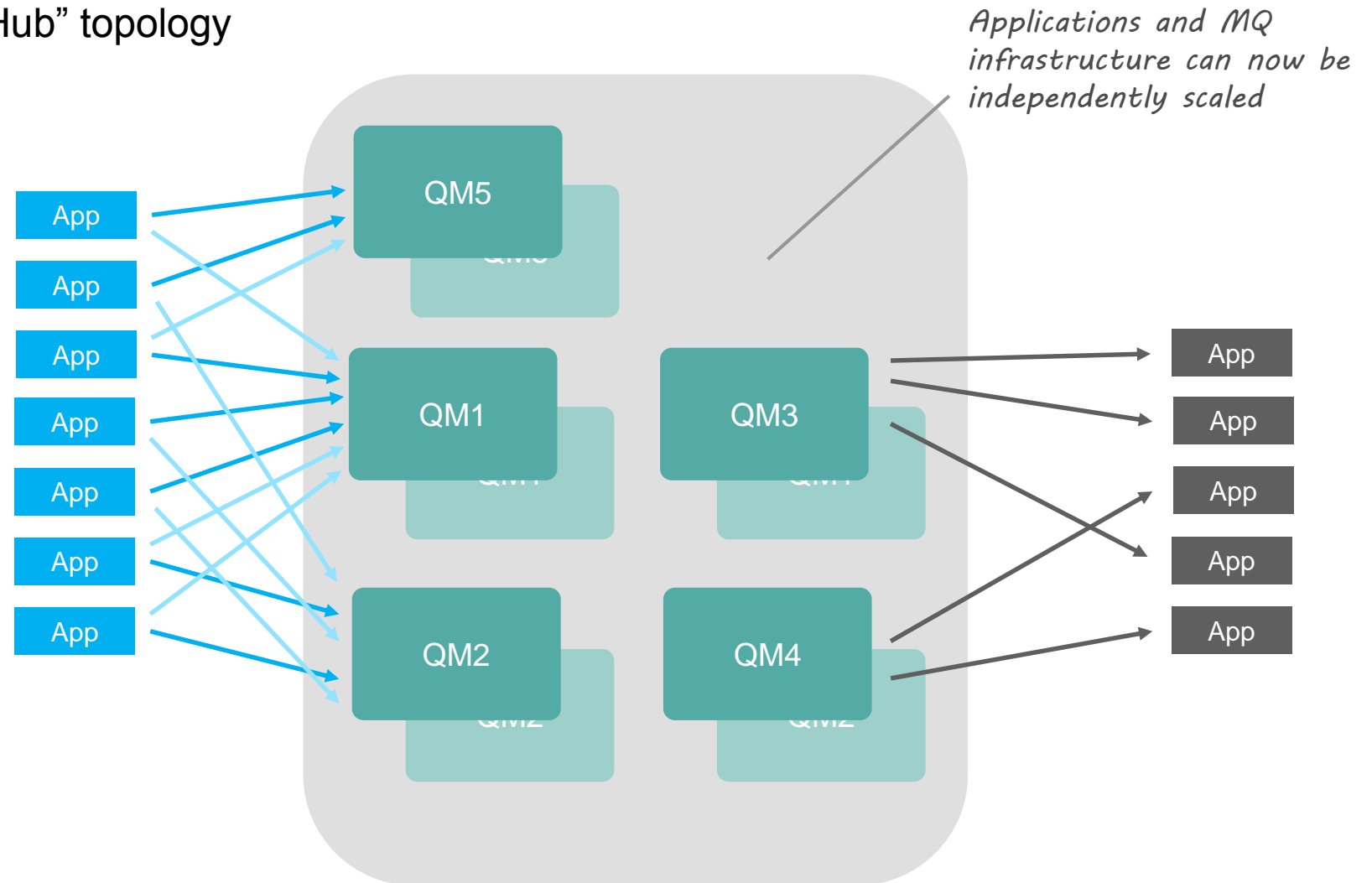


*Use an MQ cluster to workload balance and dynamically route work to the service providers*

# An MQ Hub topology

- Pulling all this together into the MQ “Hub” topology

<http://ibm.biz/MQHubPart1>

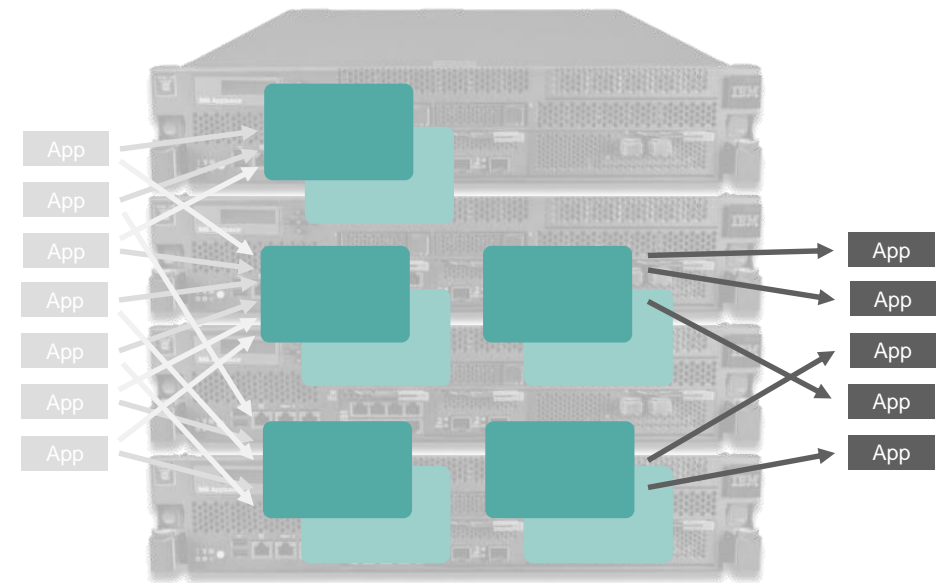


# *Runtime environments*

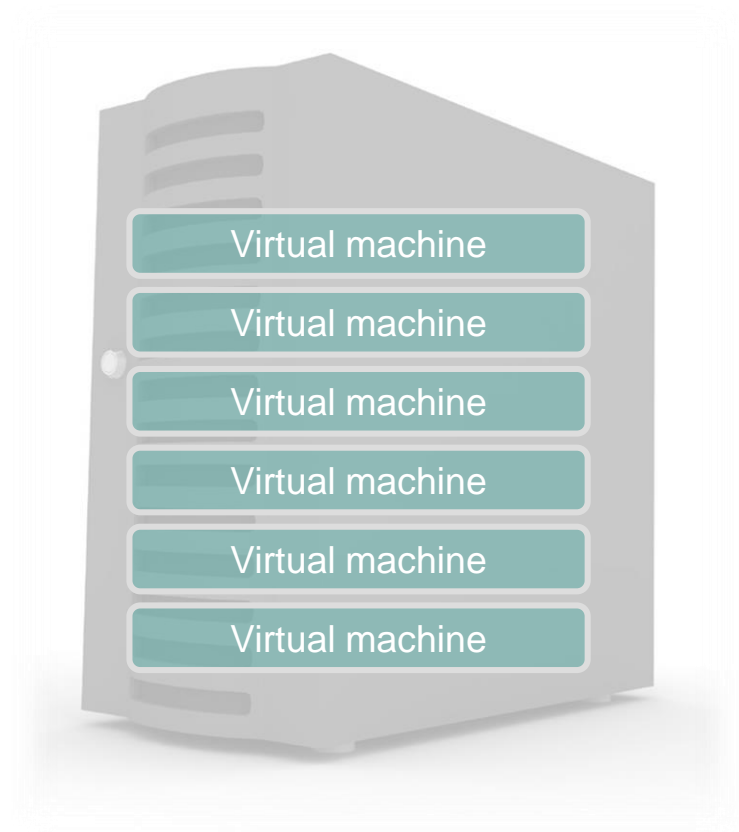
- As a service architectures apply to every type of runtime
  - ▶ Physical Hardware, Virtual machines, Containers
  - ▶ On-premise, public/private Cloud
- Management and provisioning layers can provide the real benefits
  - ▶ PureApplication
  - ▶ Docker with an orchestration layer such as Kubernetes/Swarm/...
  - ▶ Many, many others...

# Physical hardware

- IBM MQ Appliance
- A dedicated environment to easily and quickly provision and configure queue managers
- No external dependencies
  - ▶ Simplifies provisioning
  - ▶ Normalises behaviour



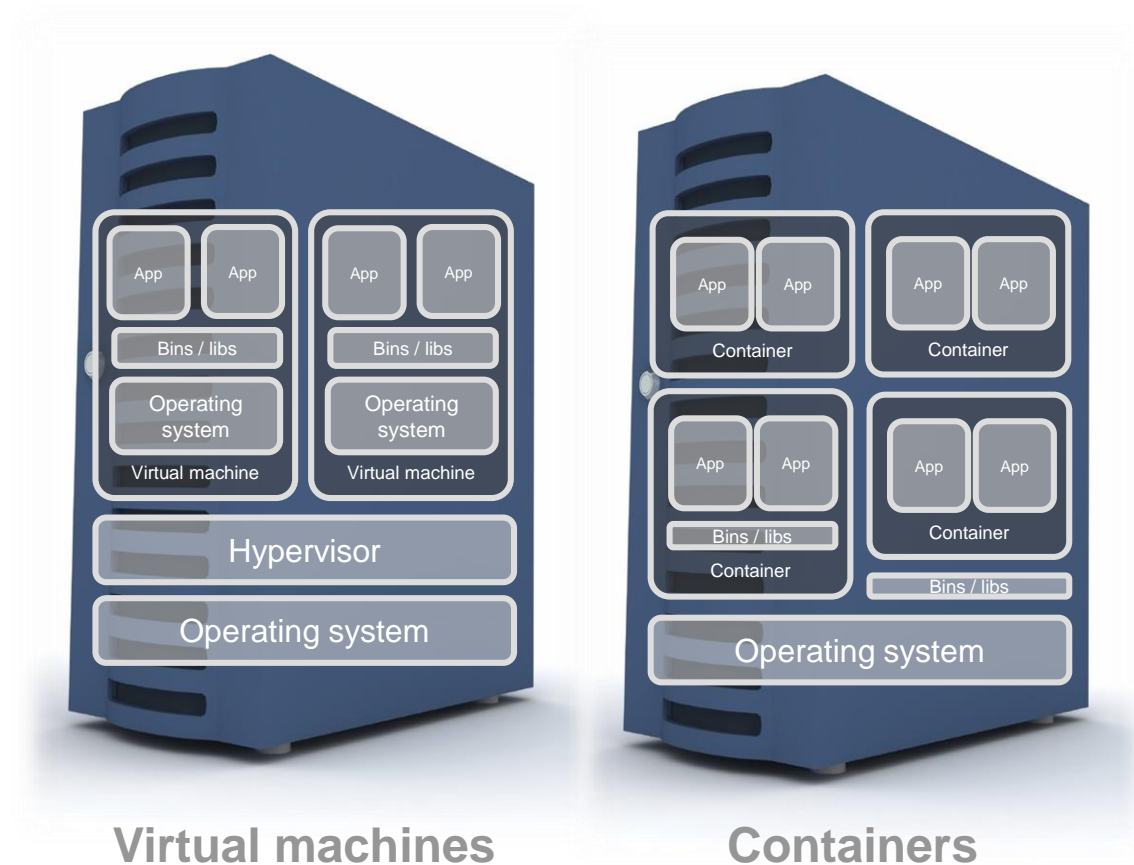
- IBM supports MQ running in virtual machines
  - ▶ Simply build your VM based on a supported MQ O/S
  - ▶ Install MQ plus any required software and configuration
  - ▶ Use and reuse, from development through to production
- Many benefits
  - ▶ Simple replication of environments
    - Using pre-canned, tried and tested, software stacks
  - ▶ Isolation from host and other VMs
    - Reduce risks from multi-tenancy
  - ▶ Separation from physical servers
    - More efficient utilisation of hardware





# Containers

- Containers provide a similar environment to a VM but lighter in weight
  - ▶ A **virtual machine** provides an abstraction of the physical hardware
  - ▶ A **container** abstracts the OS level, typically at the user level
- Linux containers
  - ▶ Containers all share the same OS kernel
  - ▶ Images are constructed from layered filesystems
  - ▶ Containers isolate applications from each other and the underlying infrastructure
- Many orchestration technologies available to manage your container environment
- IBM MQ is supported to run in **Docker** containers since MQ 8.0.0.4



Virtual machines

Containers

- Self service will increase your agility and efficiency
- Define your best practices up front
- Design your MQ architecture to be deployed as-a-service
- Apply it to the runtime and orchestration framework of your choice



***Also see IBM Messaging in the cloud***

# WHERE DO I GET MORE INFORMATION?

IBM Messaging developerWorks

[developer.ibm.com/messaging](http://developer.ibm.com/messaging)

[www.ibm.com/developerworks/community/blogs/messaging](http://www.ibm.com/developerworks/community/blogs/messaging)

IBM Messaging Youtube

<https://www.youtube.com/IBMmessagingMedia>

LinkedIn

[ibm.biz/ibmmessaging](http://ibm.biz/ibmmessaging)

Twitter

@IBMMessaging

IBM MQ Facebook

[Facebook.com/IBM-MQ-8304628654/](https://www.facebook.com/IBM-MQ-8304628654/)

