Introduction to MQ

Sam Goulden
IBM MQ L3 Service

Agenda

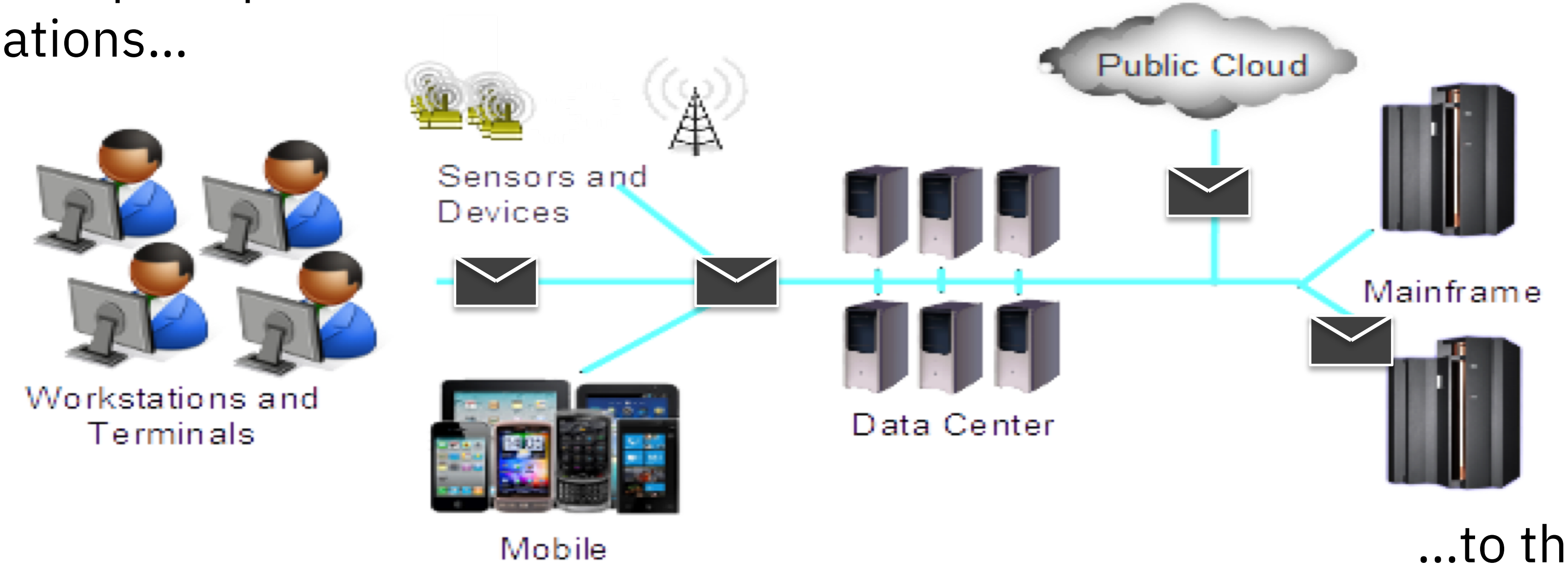
- **Messaging**
  - What is messaging and why use it?
  - What does MQ give you?

- **Fundamentals of IBM MQ**
  - Messaging models
  - Key components
  - Messaging applications
  - MQ Environments
  - Security
  - Reliability and availability
  - Administration
  - MQ Advanced

What is messaging?

- **It connects your applications!**

From the simplest pairs
of applications…

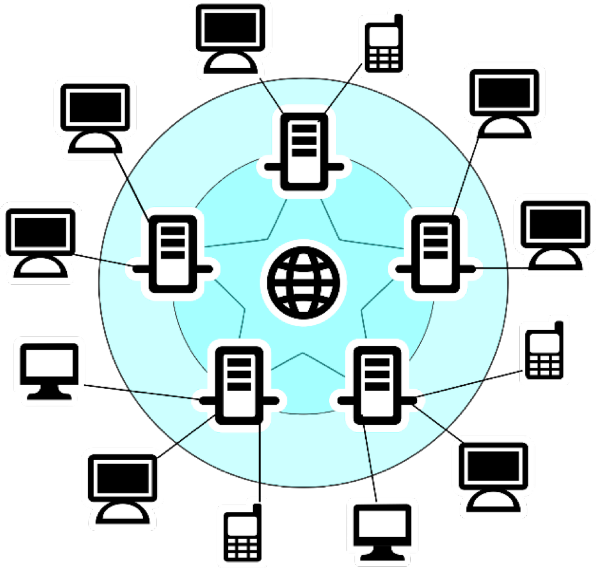


…to the most complex
business processes.

- **and breaks the tight coupling…**

Why use it?

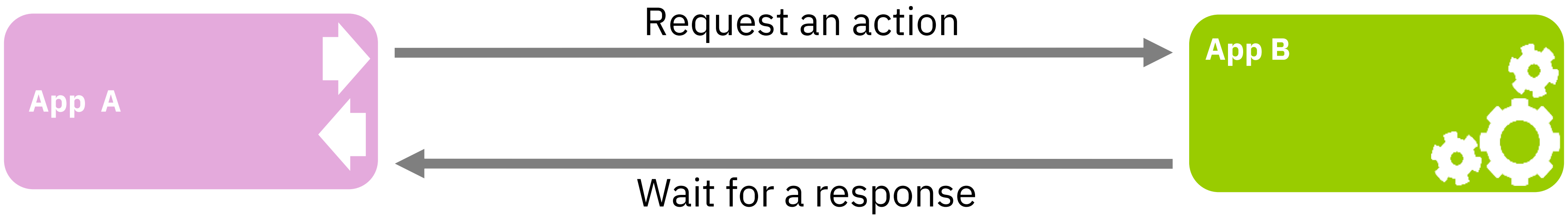
- Extended reach
- Reliability
- Scalability
- Flexibility



- Provides for simplification of application development
  - Ubiquity
  - Easy to change and scale
  - Focus on the business logic

- Important regardless of the initial scale of deployment

Direct communication between applications



- **Issues with this ‘synchronous’ approach**
  - Both applications A and B **must always** be available for A to continue
  - A cannot do anything whilst B is processing A’s request
  - What is B fails whilst A is waiting for it to complete?
  - What is B needs to handle a high workload of different priority requests?

Fragility of tight coupling

As systems become more tightly coupled, their reliance on each other increases

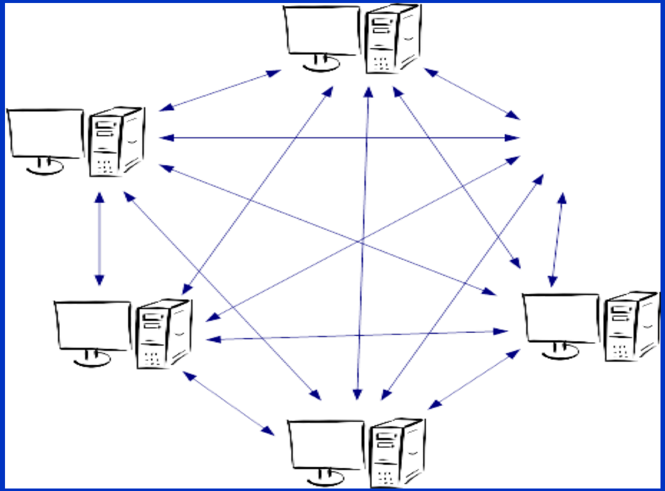
The cost of a failure of a process increases

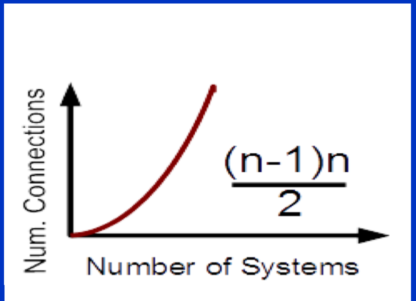
Maximum number of connections goes up with the square of
the number of systems

Scaling systems independently
to respond to requirements becomes unmanageable

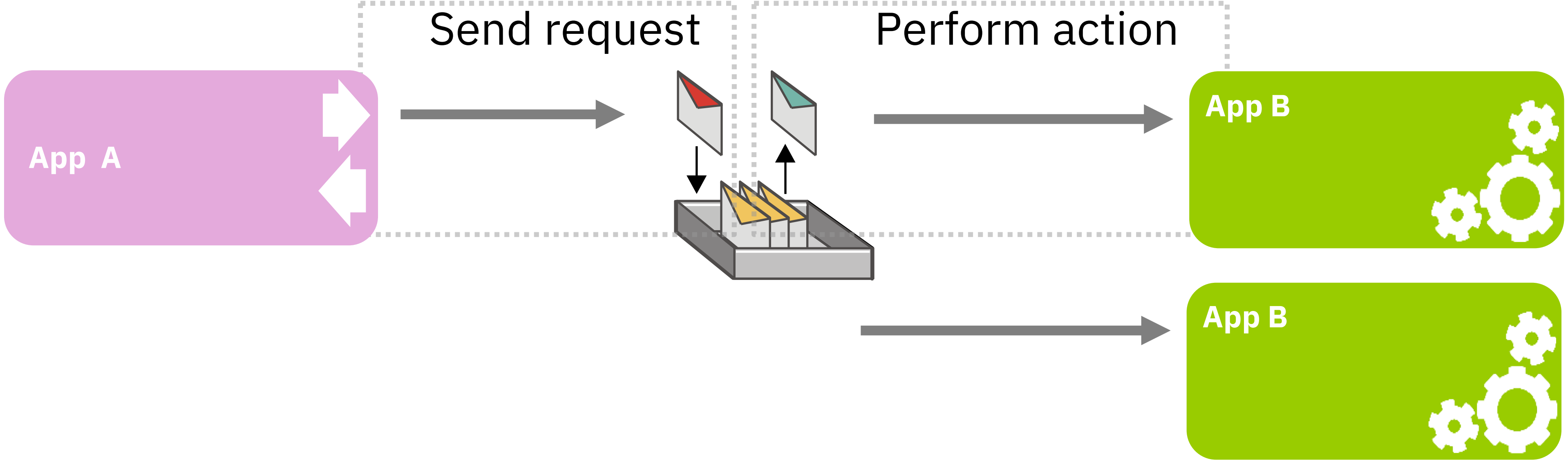
A process was originally designed for one, well-defined, purpose…it then needed to change to meet new requirements

Being able to respond rapidly to internal and external challenges by rapidly modifying existing services gives a competitive advantage



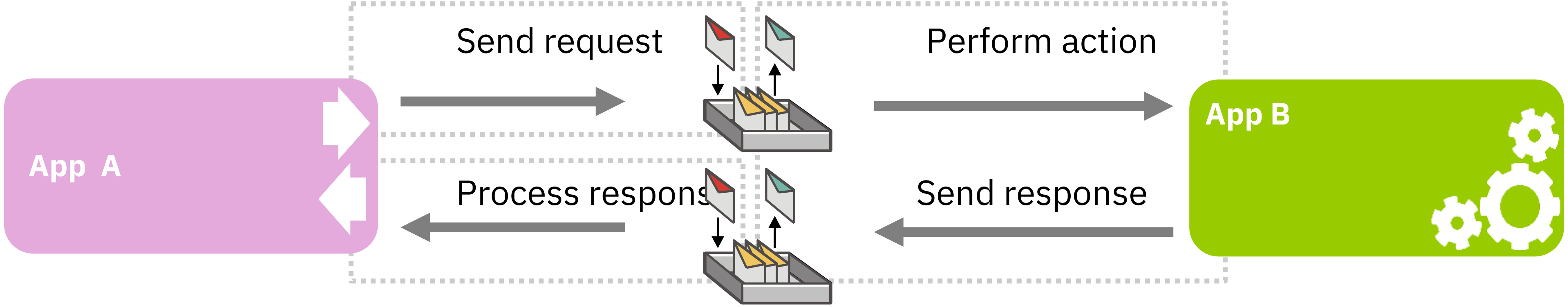


Adding flexibility with Messaging



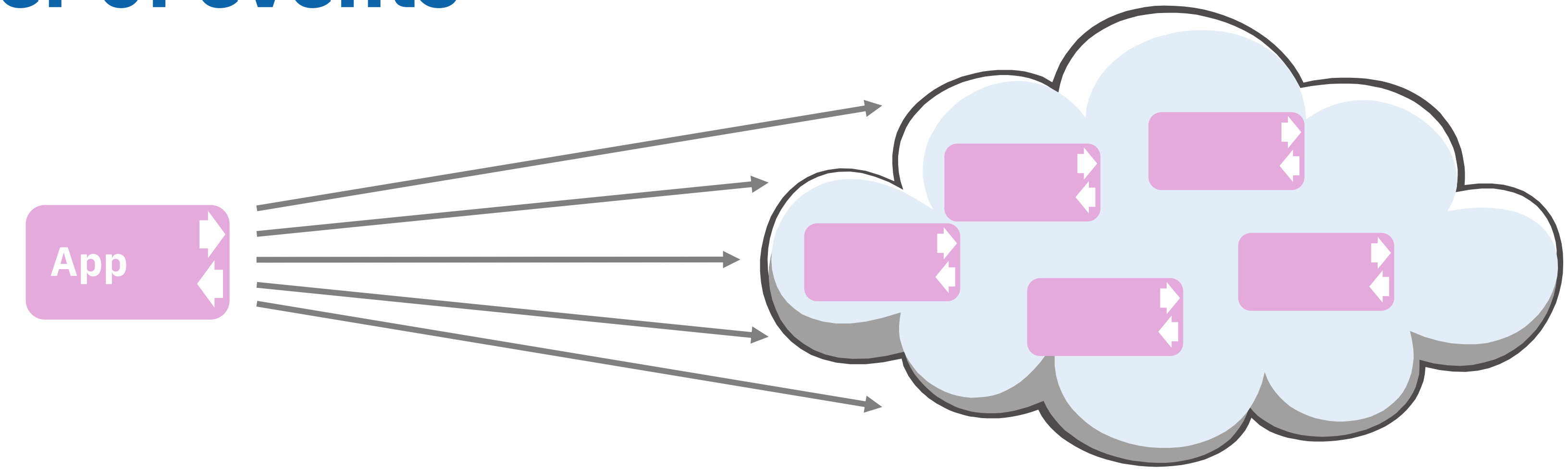
- **A ‘queue’ is placed between the two applications**
  - Allows A to continue immediately, without waiting for B
  - Allows B is efficiently process a queue of work
  - Overcomes availability of B versus A – “store and forward” of messages

What if I NEED a response?



- **Using messaging still adds value!**
  - ▶ Process the response whenever it becomes available
    - *No need for A to be idle whilst the request is performed*
  - ▶ Application B processes its workload efficiently and can handle spikes in load
  - ▶ Application, network and infrastructure failures are handled

The power of events



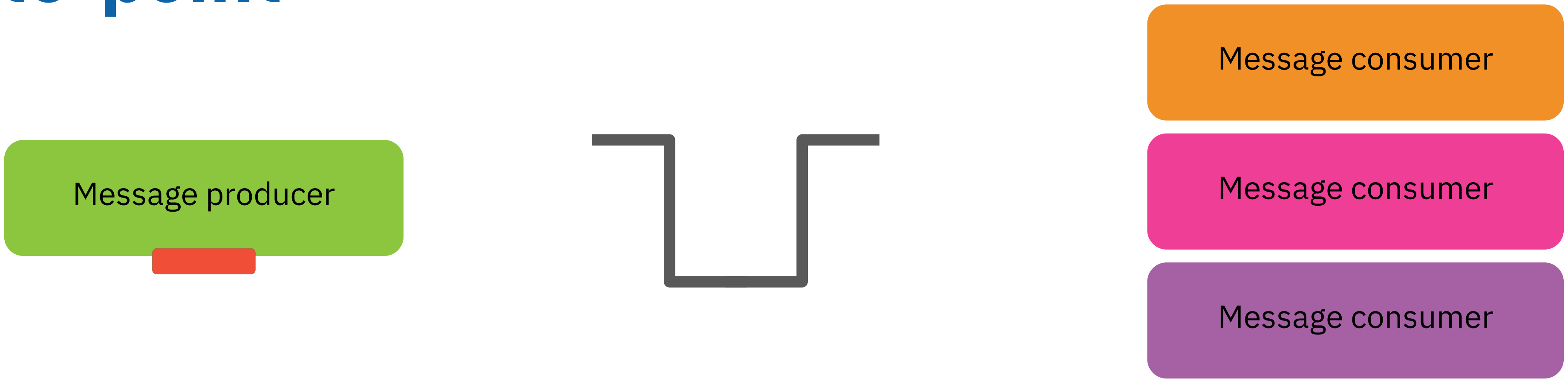
- **Not all information is distributed on a one-to-one basis**

- **Think about streams of information**
  - Regularly updated information - such as stock prices or sensor data
  - Business events - such as ‘new customer’ or ‘purchase’

- **Publish / Subscribe messaging is the solution!**
  - The owner of the information simply ***publishes*** it on a ***topic***
  - Anybody who is interested simply ***subscribes*** to the ***topic***

Messaging Models

Point-to-point



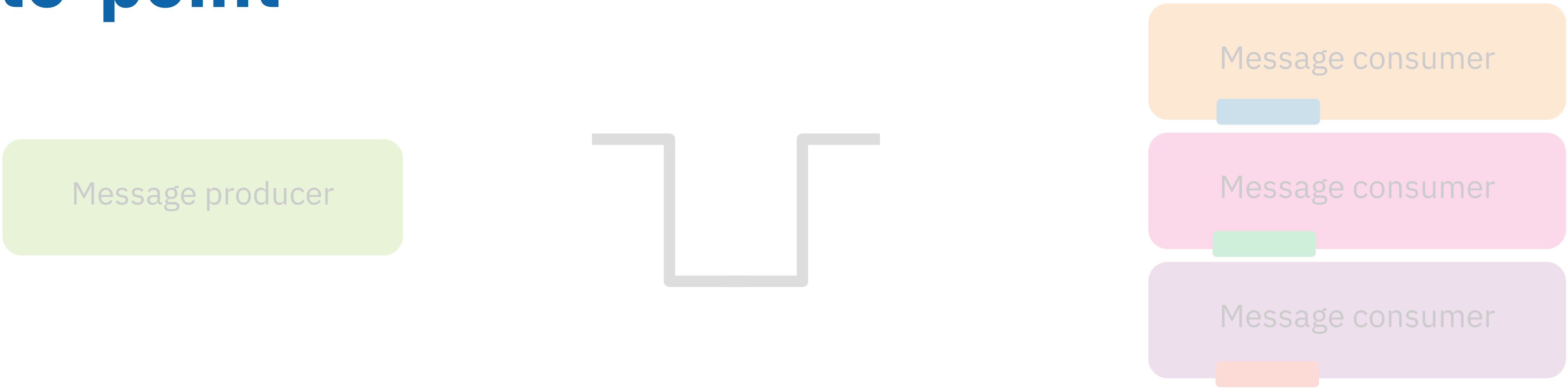
Message producer

Message consumer

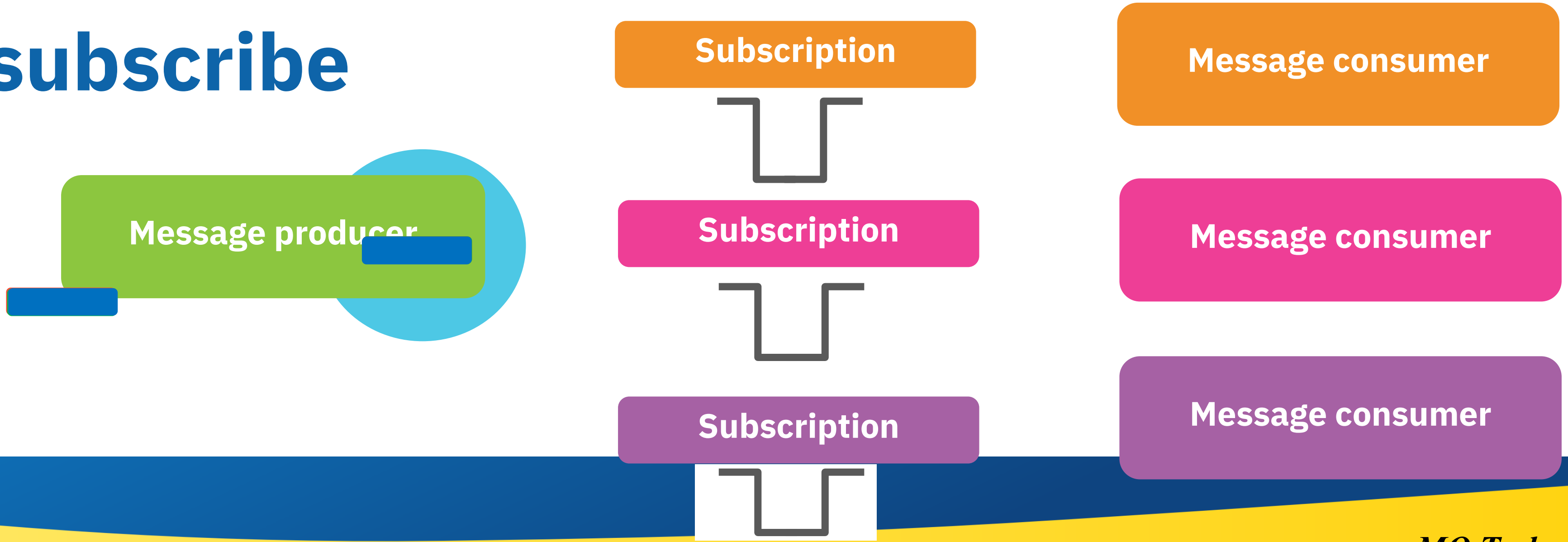
Message consumer

Message consumer

Point-to-point

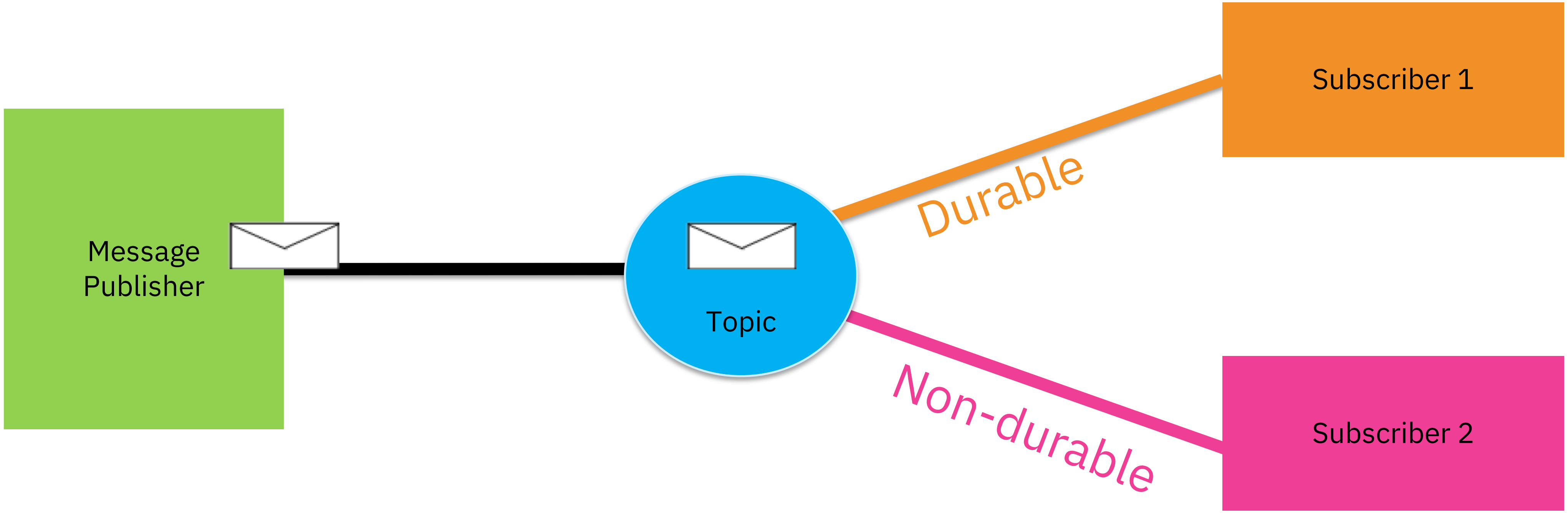


Publish/subscribe



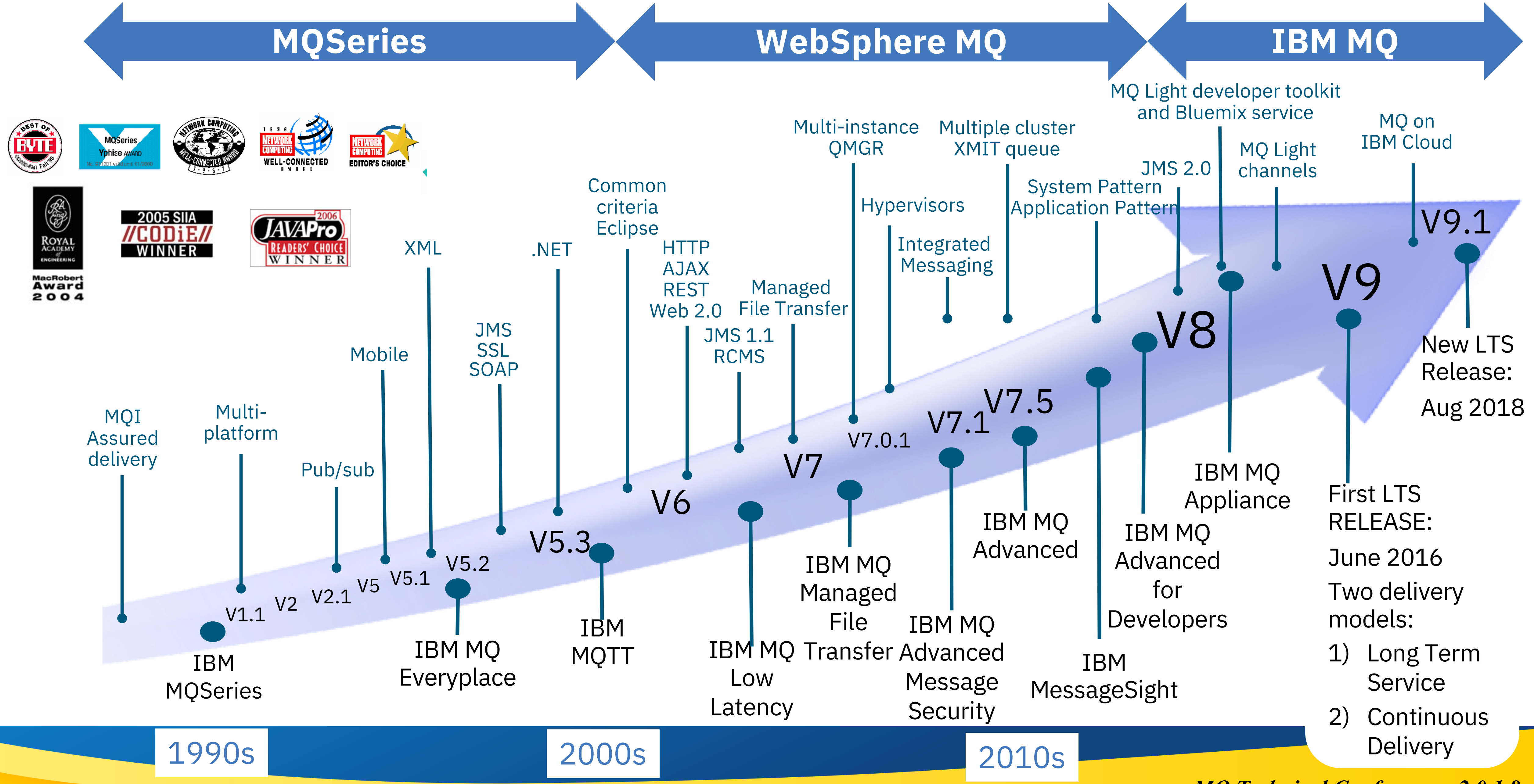
Durable publish/subscribe in action

- **Durable subscriptions result in published messages being retained when the subscriber is not connected to the messaging provider.**



IBM MQ

IBM MQ timeline



MQSeries

WebSphere MQ

IBM MQ

BEST OF BYTE
MQSeries Yphise AWARD
NETWORK COMPUTING WELL-CONNECTED AWARD
1996 NETWORK COMPUTING WELL-CONNECTED AWARD
NETWORK COMPUTING EDITOR'S CHOICE
ROYAL ACADEMY OF ENGINEERING
MacRobert Award 2004
2005 SIIA CODiE WINNER
2006 JAVAPro READERS' CHOICE WINNER

MQI Assured delivery

IBM MQSeries

Multi-platform

V1.1

V2

Pub/sub

V2.1

V5

Mobile

V5.1

XML

V5.2

JMS SSL SOAP

IBM MQ Everyplace

.NET

V5.3

IBM MQTT

Common criteria Eclipse

V6

HTTP AJAX REST Web 2.0

JMS 1.1 RCMS

IBM MQ Low Latency

Managed File Transfer

V7

IBM MQ Managed File Transfer

Multi-instance QMGR

V7.0.1

Hypervisors

Integrated Messaging

V7.1

IBM MQ Advanced Message Security

Multiple cluster XMIT queue

V7.5

IBM MQ Advanced

System Pattern Application Pattern

IBM MessageSight

JMS 2.0

V8

IBM MQ Advanced for Developers

MQ Light developer toolkit and Bluemix service

IBM MQ Appliance

MQ Light channels

V9

First LTS RELEASE:

June 2016

Two delivery models:

1) Long Term Service
2) Continuous Delivery

MQ on IBM Cloud

V9.1

New LTS Release:

Aug 2018

1990s

2000s

2010s

What MQ adds to messaging

Enterprise Messaging

Reliability

Assured message delivery “Once and once only”

Resiliency and high availability of the infrastructure

Continued support and interoperability of systems for over twenty years

Scalability

High performance solution

Incremental growth of applications and infrastructure

Ubiquity

Breadth of support for platforms and environments

Multiple application environments and APIs to suit many styles

Security

Data encryption and integrity

End use authentication and authorisation

Audit trails for configuration and data flows

Anatomy of an MQ system

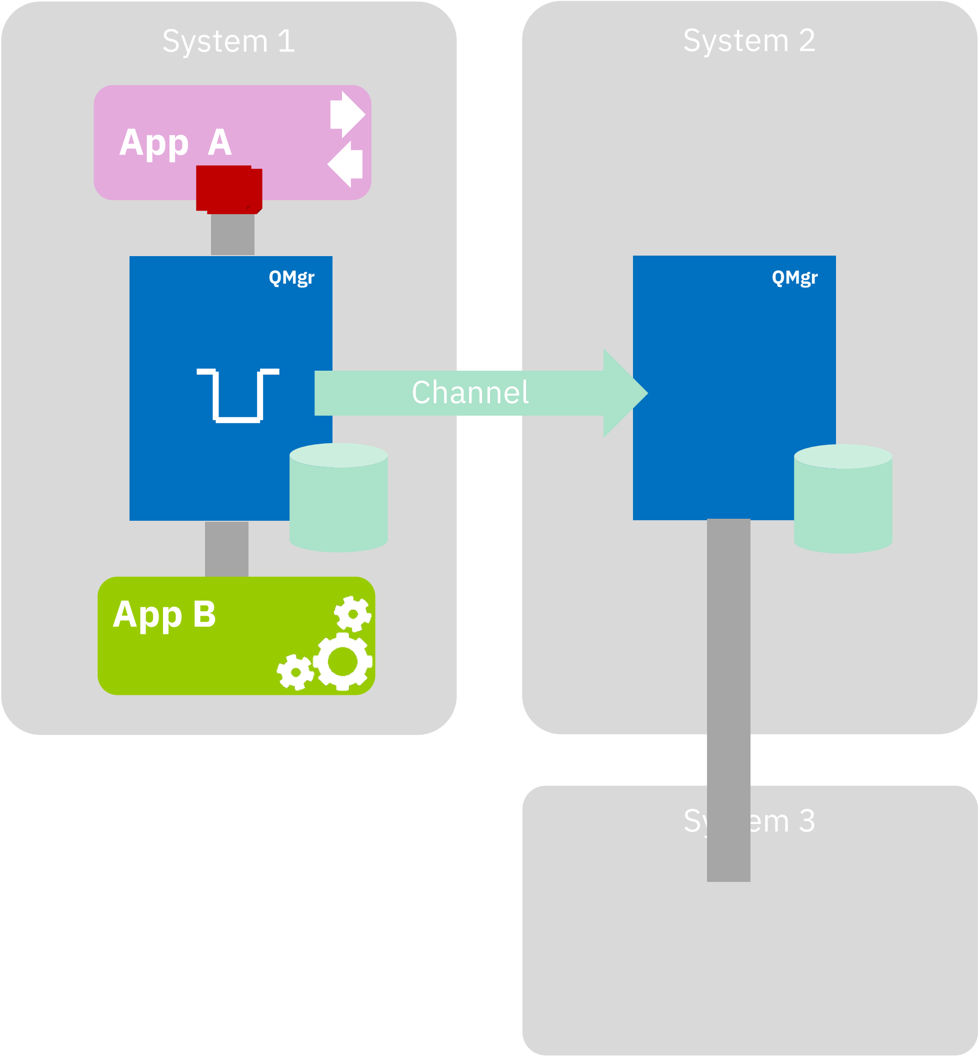
- **Applications**
  - Applications use MQ clients to connect to an MQ **queue manager**
  - Applications can connect to queue managers either on the same system (*BINDINGS mode*) or remotely over a network (*CLIENT mode*)

- **Queue Managers**
  - A queue manager is a runtime that hosts messaging resources such as **queues** and their **messages**
  - A queue manager manages the flow and storage of messages
  - Each queue manager runs on a single system
  - Multiple queue managers can be connected together using **channels** and messages routed between them

- **Queues**
  - Queues are a named resource where messages sent to by applications, stored by the queue manager and retrieved by applications

- **Messages**
  - Are just chunks of data
  - Applications build messages to send and receive

- **Channels**
  - Channels define a way for one queue manager to connect to another queue manager
  - Channels can be manually configured or dynamically created as and when needed using **MQ Clusters**

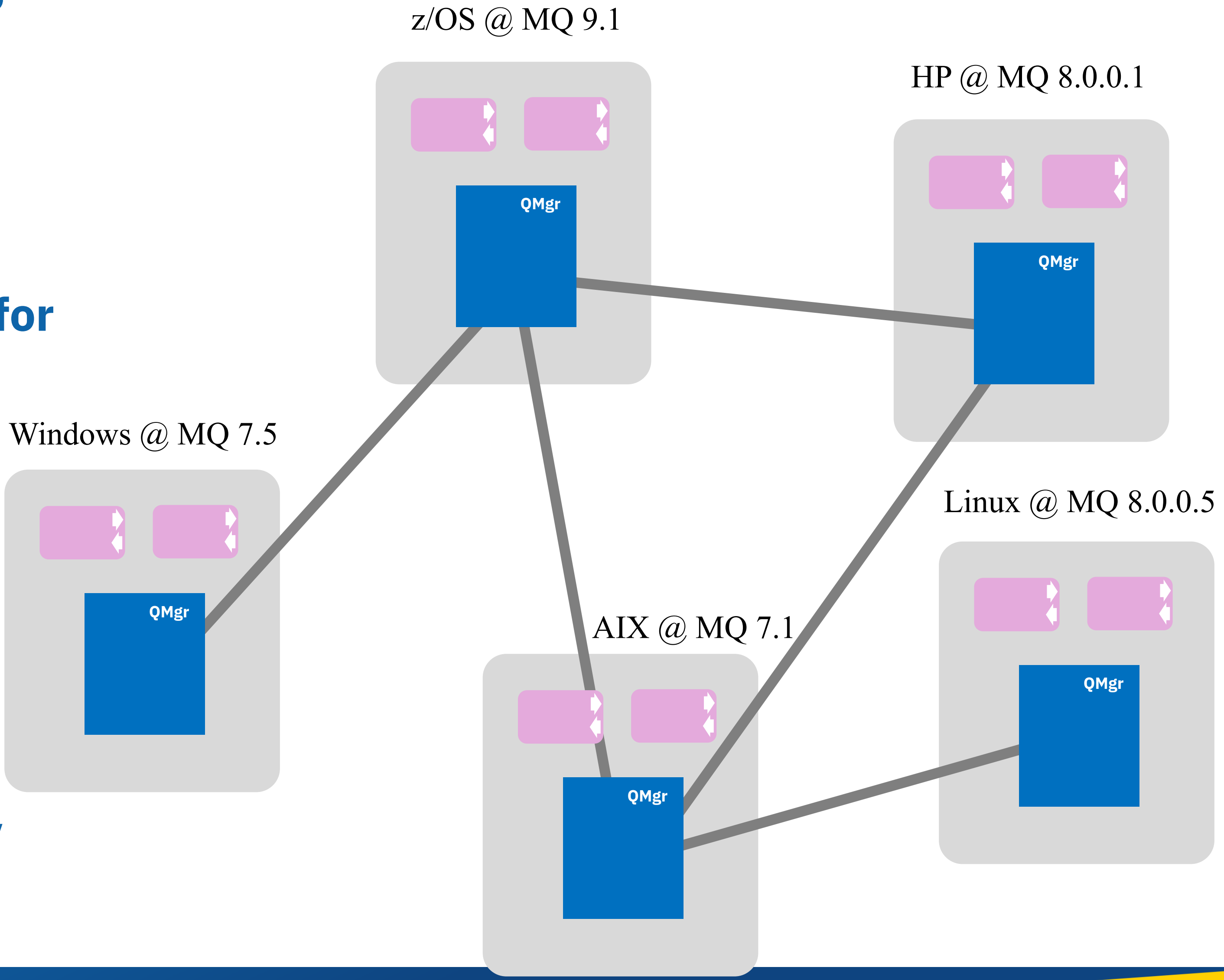


IBM MQ

IBM MQ Architecture

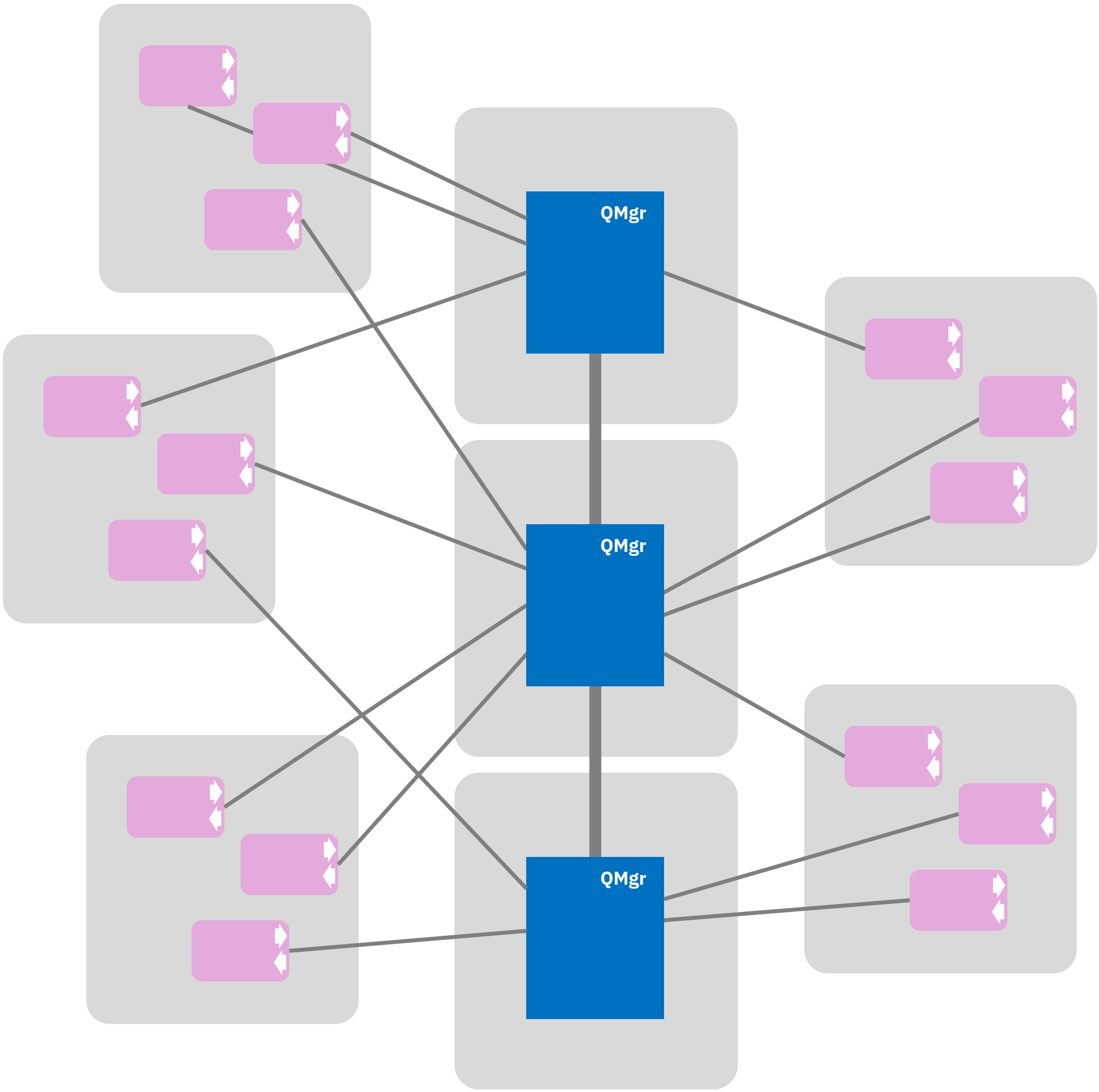
Distributed Architectures

- Used for connectivity of heterogeneous systems
- “Store and Forward” system to account for network outages
- This is the ‘original’ deployment pattern for MQ
- Queue managers will interoperate with other queue managers and clients at any other version of MQ



MQ hub architecture

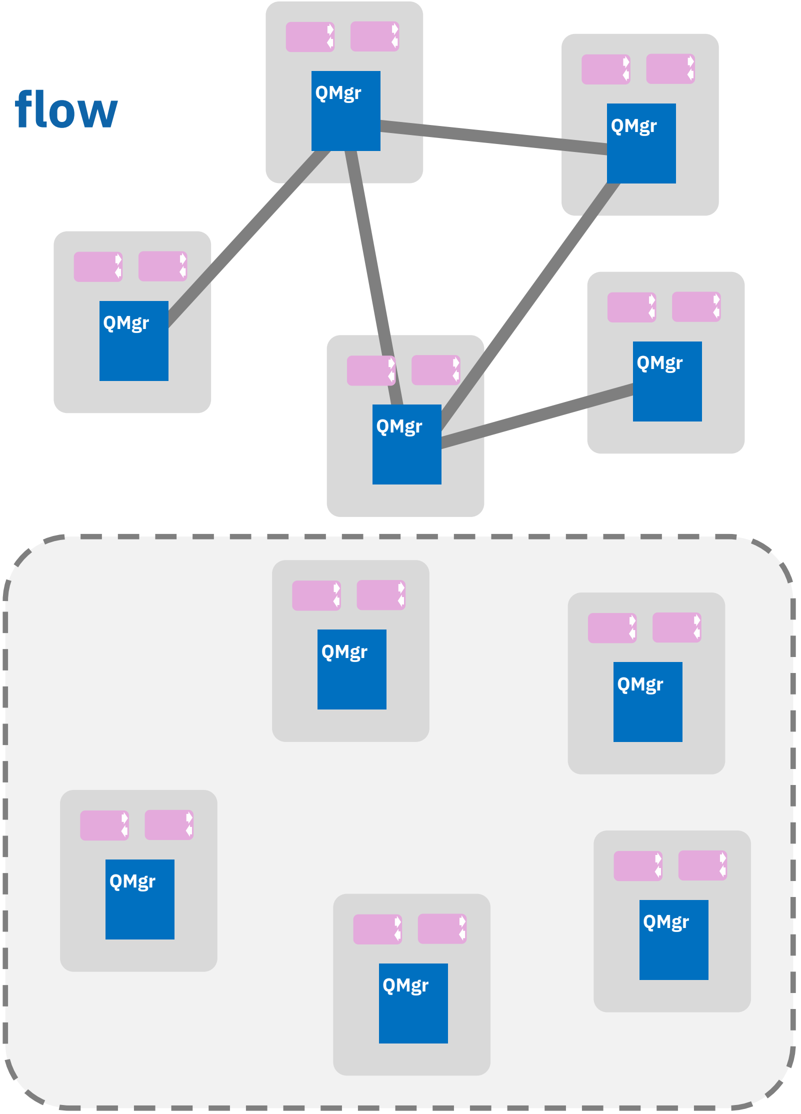
- A 'hub' of systems running queue managers on a standard deployment
- Applications connect as clients from remote systems
- This pattern has gained popularity as networks improve and administration costs go up



Connecting queue managers together

- **Channels connect queue managers together, allowing messages to flow between them**

- **Two options:**
  - **Manual configuration of channels**
    - Each channel relationship must be defined on both ends
    - Additional resource also need to be defined (***transmission queues*** and ***remote queues***)
  - **MQ clusters**
    - Once queue managers join a cluster (a pair of special channels must be defined) they can route messages to any other clustered resource in the cluster without requiring further, per queue manager, configuration.
    - As queue manager networks grow, clusters become a benefit
    - Clusters also enable workload balancing and availability routing of messages



IBM MQ

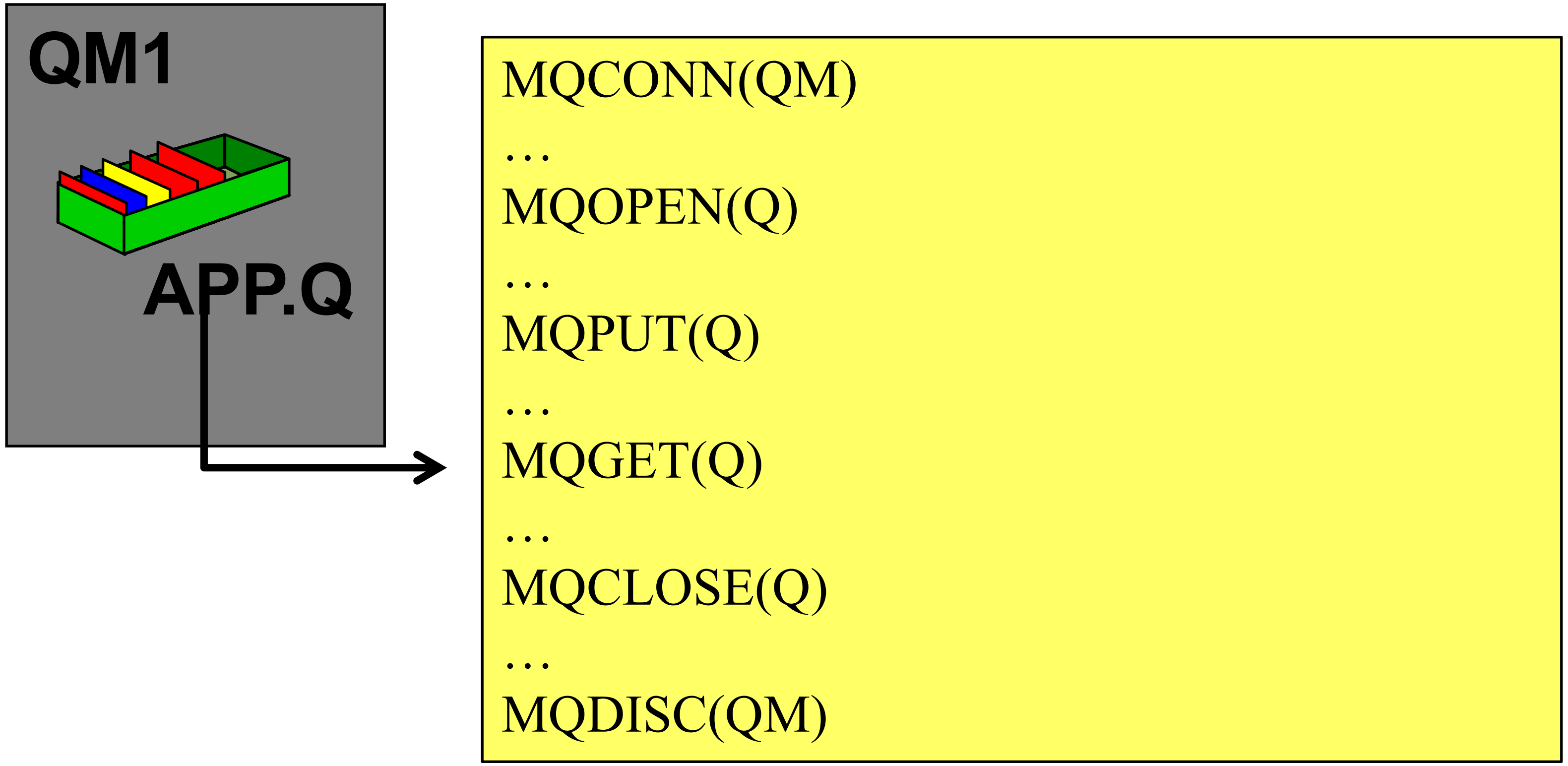
Applications

MQ APIs – How do I connect my apps to my queue manager?

- MQI
- JMS
- MQ Light API
- MQTT
- REST API Messaging (point to point only!)

MQ APIs - MQI (MQ Interface)

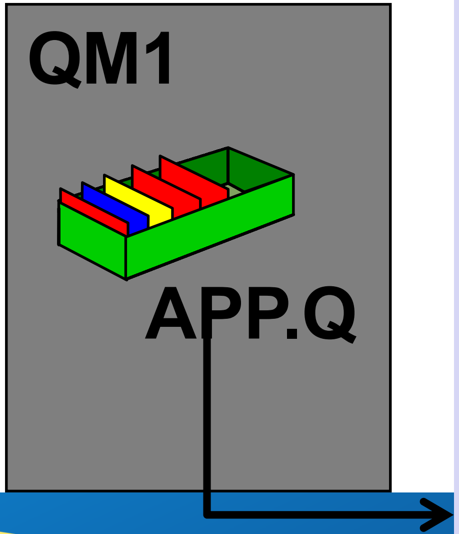
- C, COBOL, Java
- MQ's proprietary API offering full access to MQ's capabilities



```
MQCONN(QM)
…
MQOPEN(Q)
…
MQPUT(Q)
…
MQGET(Q)
…
MQCLOSE(Q)
…
MQDISC(QM)
```

MQ APIs - Java Message Service (JMS/XMS)

- **JMS is part of the JEE specification.**
  - Fully supported in application servers such as WSAS, Liberty, WebLogic and more
- **Simplifies programming for Java developers**
- **No MQ coding knowledge needed!**
- **XMS syntactically the same as JMS V1.1 but for C, C++ and C#**



```
// Lookup the MQ specific objects in JNDI
Context jndiContext         = new InitialContext();
ConnectionFactory cf        = (ConnectionFactory) jndiContext.lookup("jms/QM1");
Destination dest            = (Destination) jndiContext.lookup("jms/APP.Q");

// Establish a connection with the queue manager & create JMS objects
JMSContext context          = cf.createContext();
JMSConsumer consumer        = context.createConsumer(dest);

// Get a message
Message msg = consumer.receive();
```

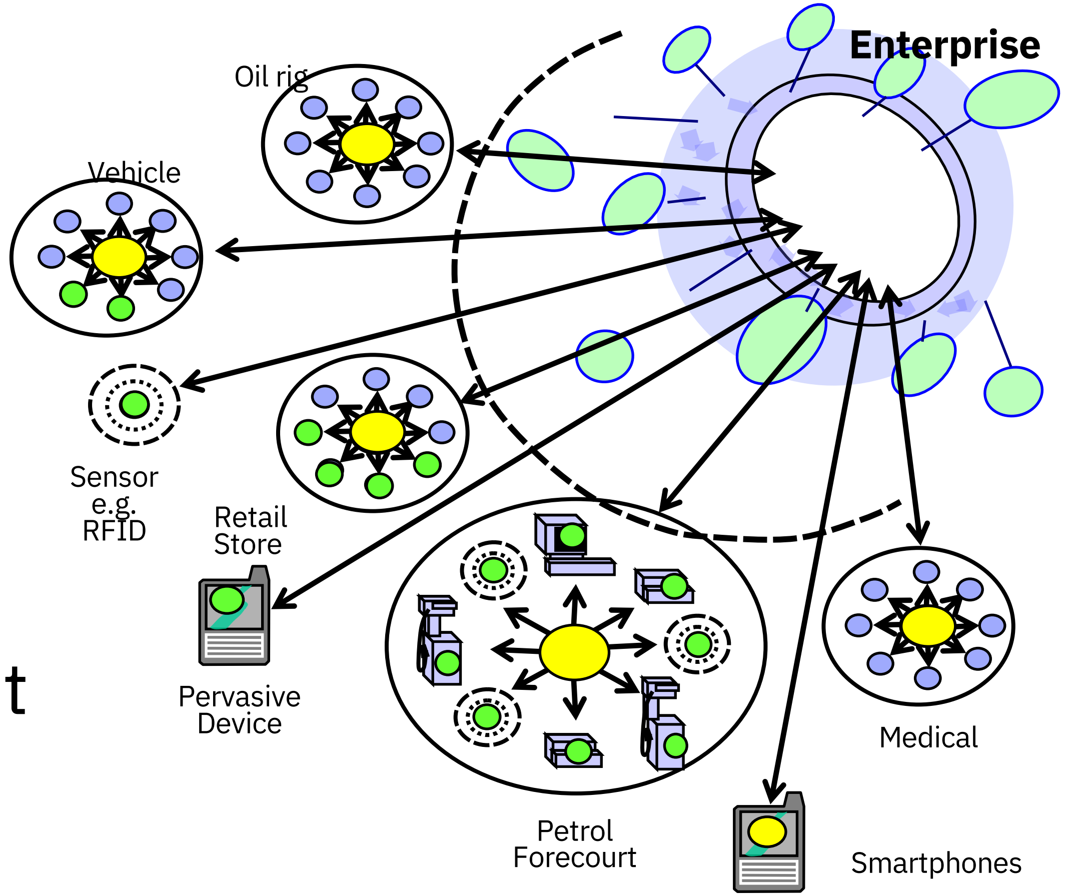
MQ APIs - MQ Light

- **AMQP based API**
- **Node.js, Java, Ruby**
- **Connects cloud applications to MQ!**

```
# Receive:
var mqlight = require('mqlight');
var recvClient = mqlight.createClient({service: 'amqp://localhost'});
recvClient.on('started', function() {
recvClient.subscribe('news/technology');
recvClient.on('message', function(data, delivery) {
          console.log(data);
     });
});
```

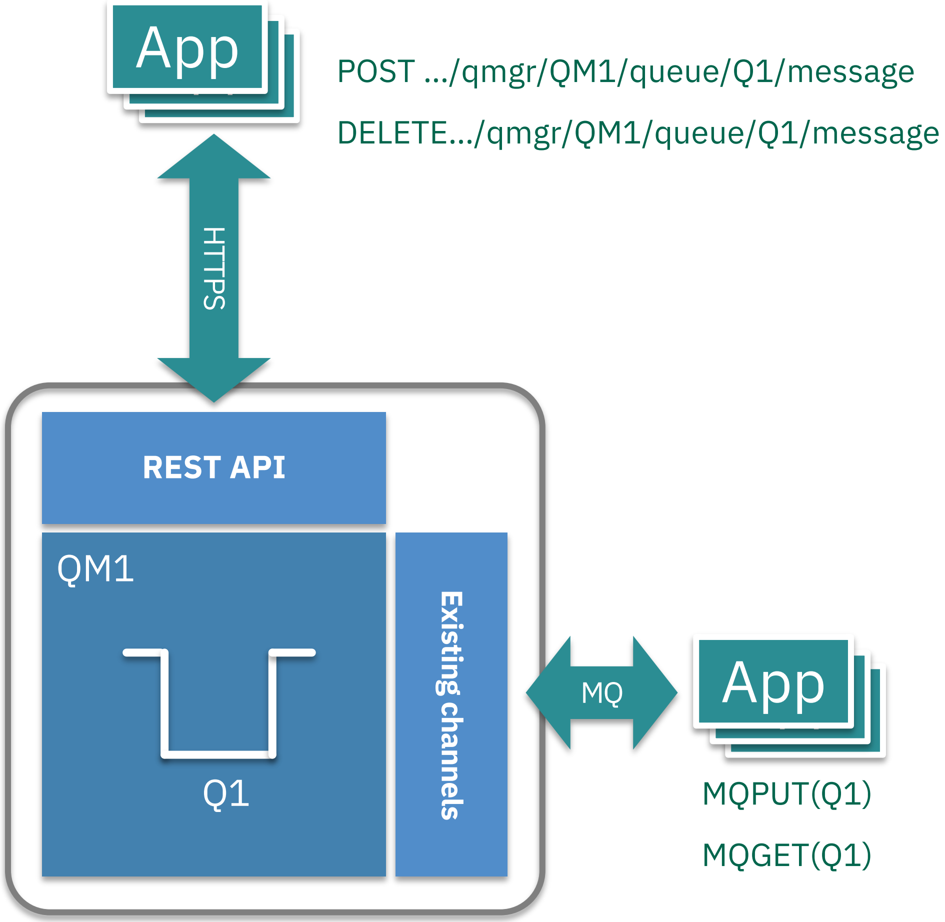
MQ APIs - MQ Telemetry (MQTT)

- Product extension supports connectivity for smart devices to the enterprise
- Utilises the open standard MQTT protocol
  - a lightweight, public, low bandwidth messaging protocol for scenarios where enterprise messaging clients are too big or bandwidth intensive.
- Java, C and JavaScript libraries provided, but you can “roll your own” that implement the MQTT v3 spec



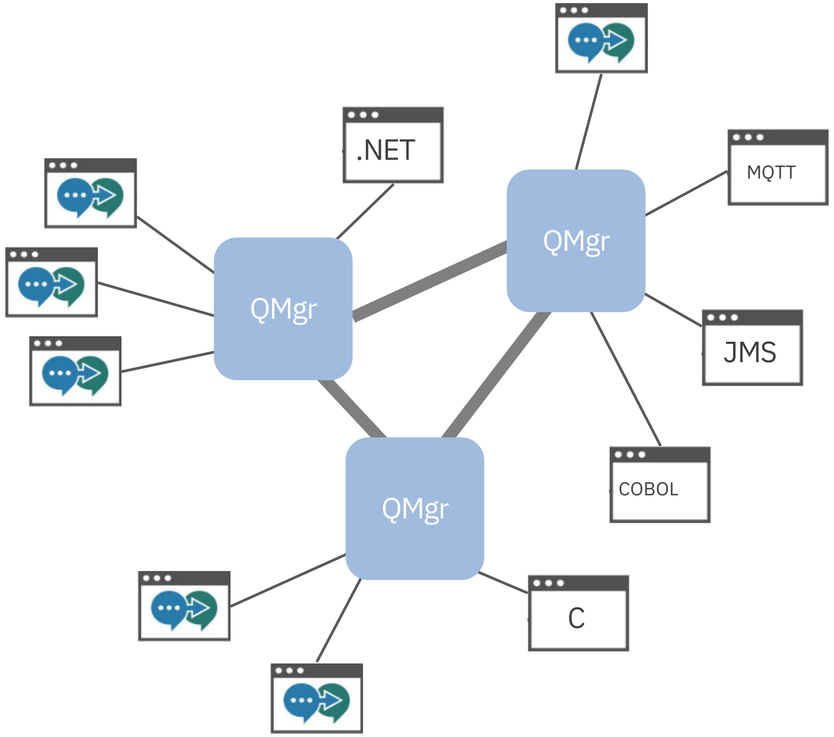
REST Messaging

- **The new HTTP server support in MQ 9.0.x provides the platform for a properly integrated REST API solution**
- **Allowing applications to put and get messages from a queue without installing any MQ software locally**
- **Ideal for environments with native REST support, such as common JavaScript libraries including NodeJS, and AngularJS**
- **Can only be used for point to point messaging**
- **For full functionality and resiliency an MQ client should still be used**


App
POST .../qmgr/QM1/queue/Q1/message
DELETE.../qmgr/QM1/queue/Q1/message
HTTPS
REST API
QM1
Q1
Existing channels
MQ
App
MQPUT(Q1)
MQGET(Q1)

Messaging APIs

- All interoperate with each other!
  - Any application can receive messages from any other application

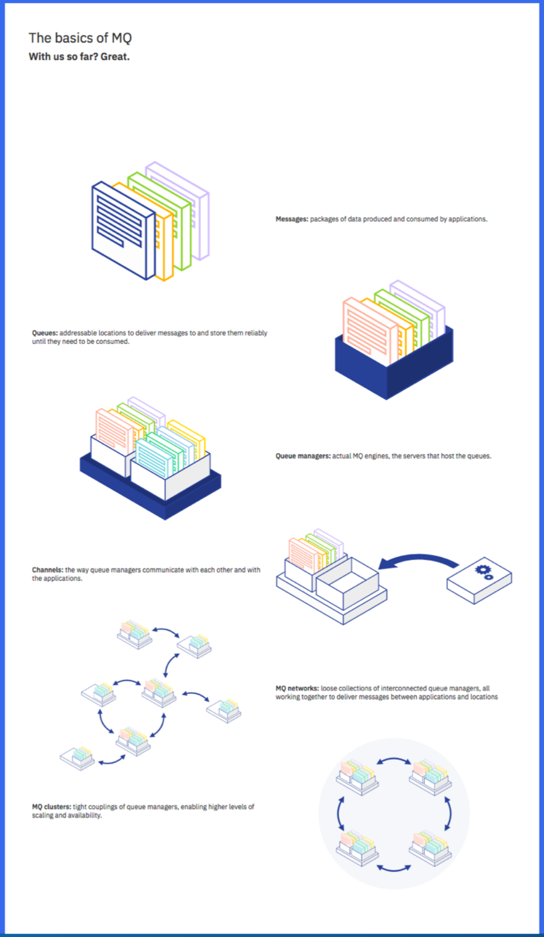


- LearnMQ

- Finding it hard to get developers started with MQ?

- Point them to: developer.ibm.com/messaging/learn-mq

- Totally new to MQ? Learn the basics



The basics of MQ

With us so far? Great.

Messages: packages of data produced and consumed by applications.

Queues: addressable locations to deliver messages to and store them reliably until they need to be consumed.

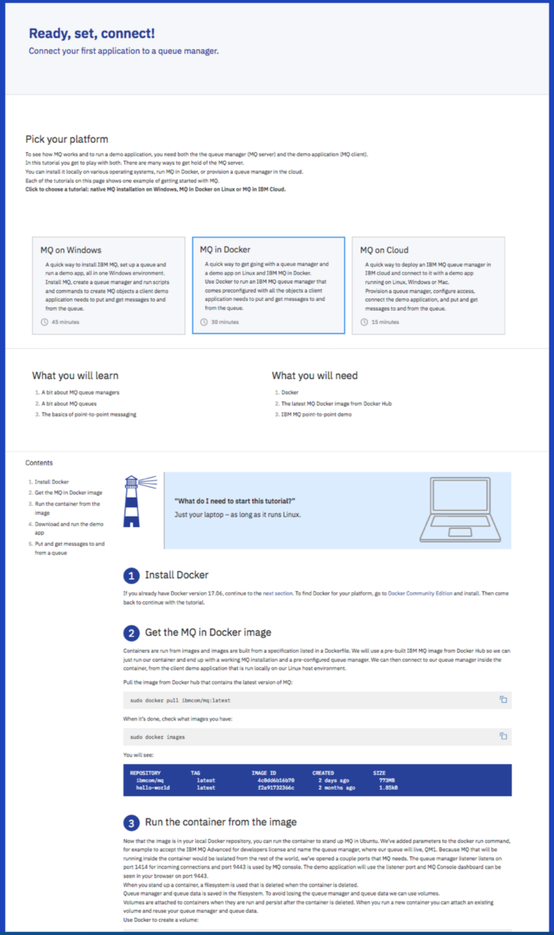
Queue managers: actual MQ engines, the servers that host the queues.

Channels: the way queue managers communicate with each other and with the applications.

MQ networks: loose collections of interconnected queue managers, all working together to deliver messages between applications and locations

MQ clusters: tight couplings of queue managers, enabling higher levels of scaling and availability.

- Step-by-step guide to getting up and running with MQ



Ready, set, connect!

Connect your first application to a queue manager.

Pick your platform

To see how MQ works and to run a demo application, you need both the the queue manager (MQ server) and the demo application (MQ client).
In this tutorial you get to play with both. There are many ways to get hold of the MQ server.
You can install it locally on various operating systems, run MQ in Docker, or provision a queue manager in the cloud.
Each of the tutorials on this page shows one example of getting started with MQ.
Click to choose a tutorial: native MQ installation on Windows, MQ in Docker on Linux or MQ in IBM Cloud.

MQ on Windows
A quick way to install IBM MQ, set up a queue and run a demo app, all in one Windows environment.
Install MQ, create a queue manager and run scripts and commands to create MQ objects a client demo application needs to put and get messages to and from the queue.
45 minutes

MQ in Docker
A quick way to get going with a queue manager and a demo app on Linux and IBM MQ in Docker.
Use Docker to run an IBM MQ queue manager that comes preconfigured with all the objects a client application needs to put and get messages to and from the queue.
30 minutes

MQ on Cloud
A quick way to deploy an IBM MQ queue manager in IBM cloud and connect to it with a demo app running on Linux, Windows or Mac.
Provision a queue manager, configure access, connect the demo application, and put and get messages to and from the queue.
15 minutes

What you will learn

1. A bit about MQ queue managers
2. A bit about MQ queues
3. The basics of point-to-point messaging

What you will need

1. Docker
2. The latest MQ Docker image from Docker Hub
3. IBM MQ point-to-point demo

Contents

1. Install Docker
2. Get the MQ in Docker image
3. Run the container from the image
4. Download and run the demo app
5. Put and get messages to and from a queue

"What do I need to start this tutorial?"

Just your laptop – as long as it runs Linux.

1 Install Docker

If you already have Docker version 17.06, continue to the next section. To find Docker for your platform, go to Docker Community Edition and install. Then come back to continue with the tutorial.

2 Get the MQ in Docker image

Containers are run from images and images are built from a specification listed in a Dockerfile. We will use a pre-built IBM MQ image from Docker Hub so we can just run our container and end up with a working MQ installation and a pre-configured queue manager. We can then connect to our queue manager inside the container, from the client demo application that is run locally on our Linux host environment.

Pull the image from Docker hub that contains the latest version of MQ:

```
sudo docker pull ibmcom/mq:latest
```

When it's done, check what images you have:

```
sudo docker images
```

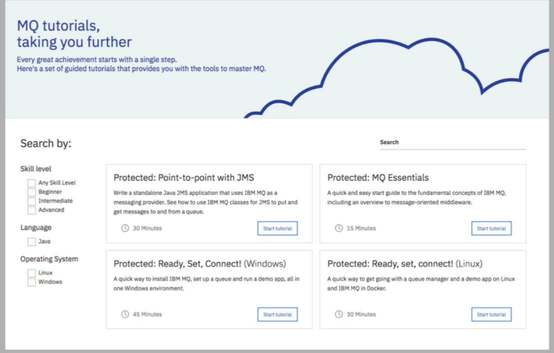
You will see:

```
REPOSITORY        TAG          IMAGE ID          CREATED          SIZE
  ibmcom/mq         latest       4c0dd6b16b70      2 days ago       773MB
  hello-world       latest       f2a91732366c      2 months ago     1.85kB
```

3 Run the container from the image

Now that the image is in your local Docker repository, you can run the container to stand up MQ in Ubuntu. We've added parameters to the docker run command, for example to accept the IBM MQ Advanced for developers license and name the queue manager, where our queue will live, QM1. Because MQ that will be running inside the container would be isolated from the rest of the world, we've opened a couple ports that MQ needs. The queue manager listener listens on port 1414 for incoming connections and port 9443 is used by MQ console. The demo application will use the listener port and MQ Console dashboard can be seen in your browser on port 9443.
When you stand up a container, a filesystem is used that is deleted when the container is deleted.
Queue manager and queue data is saved in the filesystem. To avoid losing the queue manager and queue data we can use volumes.
Volumes are attached to containers when they are run and persist after the container is deleted. When you run a new container you can attach an existing volume and reuse your queue manager and queue data.
Use Docker to create a volume:

- Tutorials on building your applications



MQ tutorials,
taking you further

Every great achievement starts with a single step.
Here's a set of guided tutorials that provides you with the tools to master MQ.

Search by:

Search

Skill level
□ Any Skill Level
□ Beginner
□ Intermediate
□ Advanced

Language
□ Java

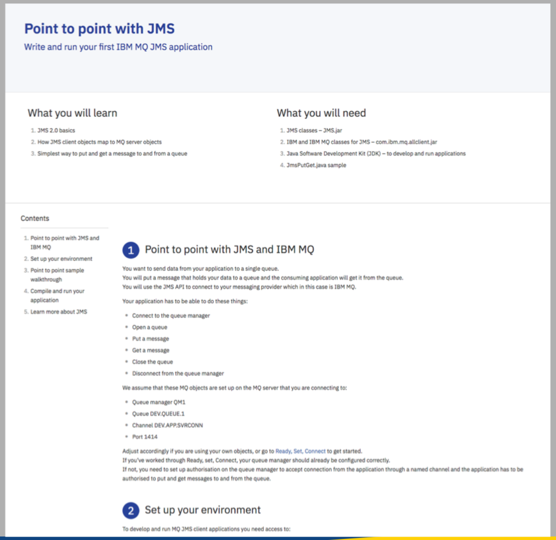
Operating System
□ Linux
□ Windows

Protected: Point-to-point with JMS
Write a standalone Java JMS application that uses IBM MQ as a messaging provider. See how to use IBM MQ classes for JMS to put and get messages to and from a queue.
30 Minutes
Start tutorial

Protected: MQ Essentials
A quick and easy start guide to the fundamental concepts of IBM MQ, including an overview to message-oriented middleware.
15 Minutes
Start tutorial

Protected: Ready, Set, Connect! (Windows)
A quick way to install IBM MQ, set up a queue and run a demo app, all in one Windows environment.
45 Minutes
Start tutorial

Protected: Ready, set, connect! (Linux)
A quick way to get going with a queue manager and a demo app on Linux and IBM MQ in Docker.
30 Minutes
Start tutorial



Point to point with JMS

Write and run your first IBM MQ JMS application

What you will learn

1. JMS 2.0 basics
2. How JMS client objects map to MQ server objects
3. Simplest way to put and get a message to and from a queue

What you will need

1. JMS classes – JMS.jar
2. IBM and IBM MQ classes for JMS – com.ibm.mq.allclient.jar
3. Java Software Development Kit (JDK) – to develop and run applications
4. JmsPutGet.java sample

Contents

1. Point to point with JMS and IBM MQ
2. Set up your environment
3. Point to point sample walkthrough
4. Compile and run your application
5. Learn more about JMS

1 Point to point with JMS and IBM MQ

You want to send data from your application to a single queue.
You will put a message that holds your data to a queue and the consuming application will get it from the queue.
You will use the JMS API to connect to your messaging provider which in this case is IBM MQ.

Your application has to be able to do these things:

- Connect to the queue manager
- Open a queue
- Put a message
- Get a message
- Close the queue
- Disconnect from the queue manager

We assume that these MQ objects are set up on the MQ server that you are connecting to:

- Queue manager QM1
- Queue DEV.QUEUE.1
- Channel DEV.APP.SVRCONN
- Port 1414

Adjust accordingly if you are using your own objects, or go to Ready, Set, Connect to get started.
If you've worked through Ready, set, Connect, your queue manager should already be configured correctly.
If not, you need to set up authorisation on the queue manager to accept connection from the application through a named channel and the application has to be authorised to put and get messages to and from the queue.

2 Set up your environment

To develop and run MQ JMS client applications you need access to:

“Once and once only delivery”

- **Message persistence**
  - **Persistent messages**
    - Stored to disk
    - Queued messages are recovered following a server failure
      - No matter what the failure, as long as the disks are intact, so will your messages be
  - **Non-persistent messages**
    - Kept in memory as much as possible (better performance)
    - Queued messages are lost in the event of a server failure or restart

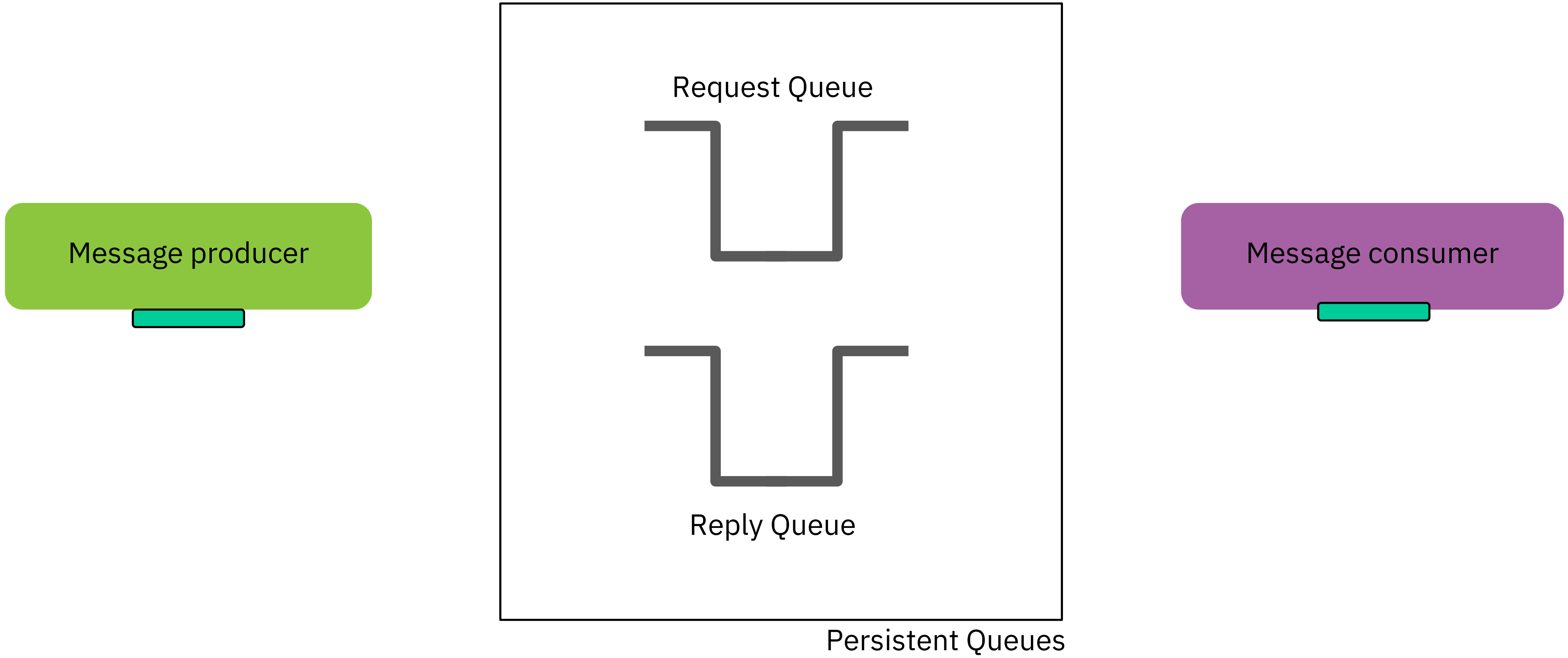
- **Transactions**
  - Multiple messaging operations can be coordinated as a transaction
  - Messaging applications are often updating other resources based on messages
    - E.g. Receive a message, insert the data to a database
  - MQ applications can coordinate messaging operations with other transactional resources
    - A queue manager can be an XA transaction coordinator
    - Or coordinated externally, for example a JEE application server such as WebSphere Application Server
  - Available in MQI, JMS and XMS APIs

- ***Combining persistent messages with transactions gives you once and once only delivery of messages from an application’s point of view***

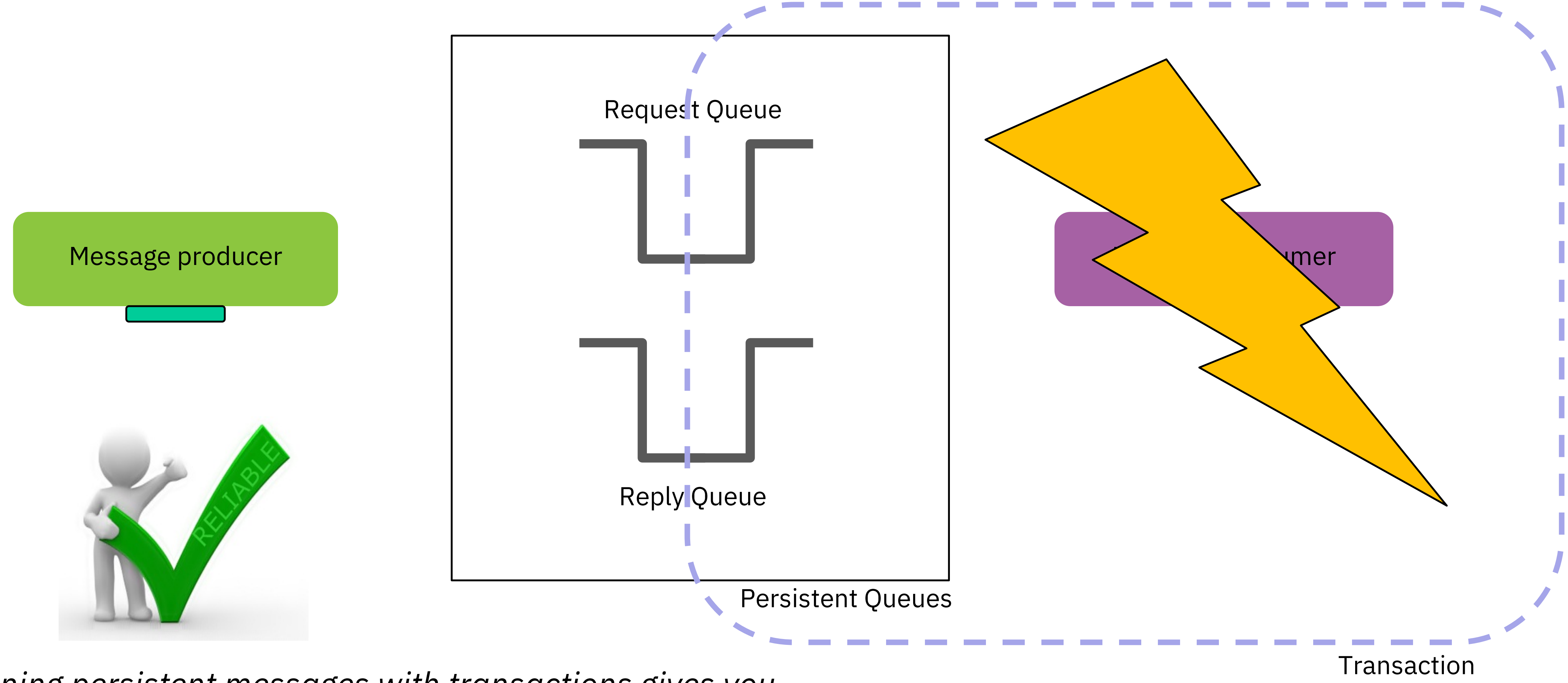
Transactional Messaging

- Non Persistent

- Persistent



Transactional Messaging



Combining persistent messages with transactions gives you once and once only delivery of messages from an application's point of view

IBM MQ
Environments
On-premise & Cloud

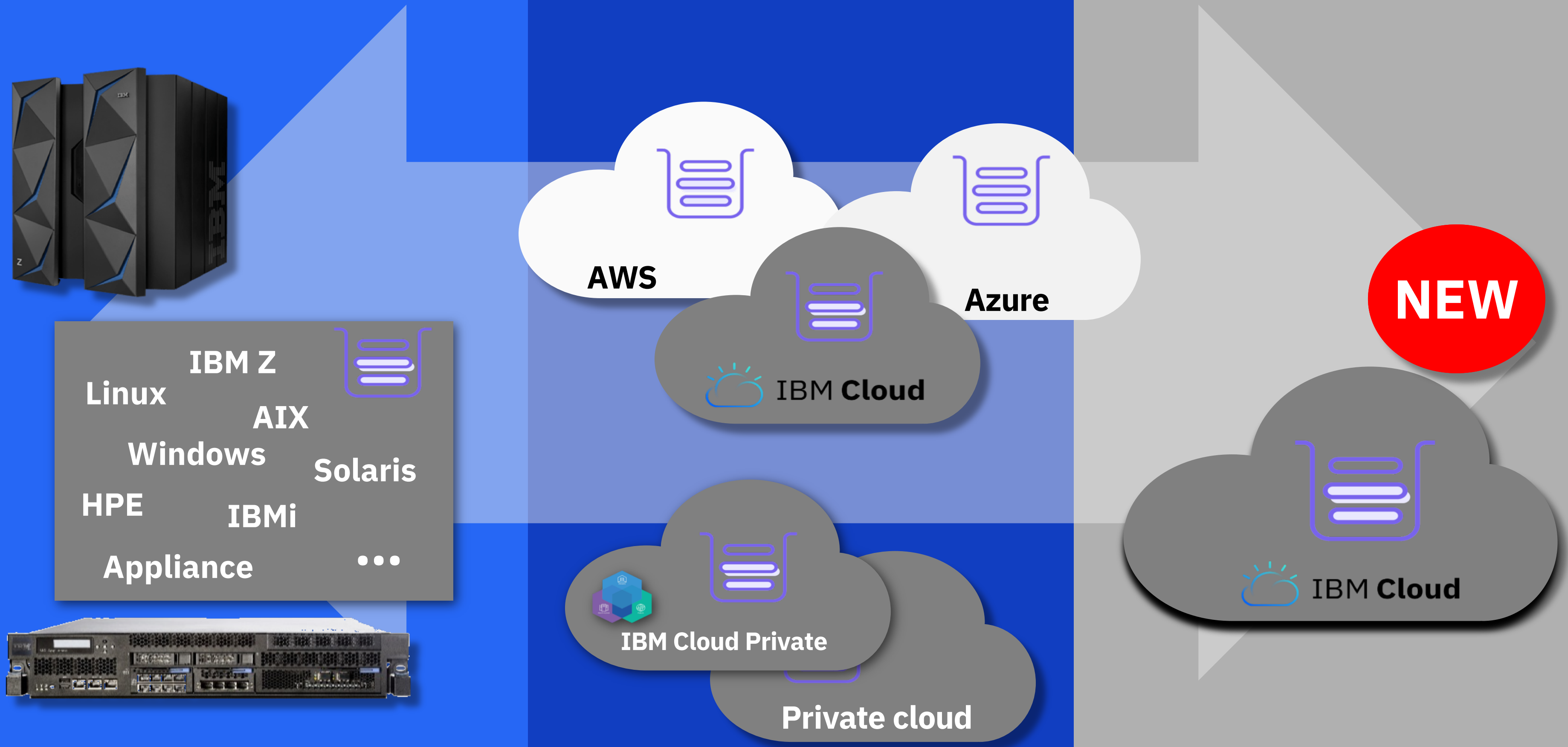
- Run IBM MQ in any location or cloud exactly as you need it

IBM MQ

- On-premise, software and the MQ Appliance

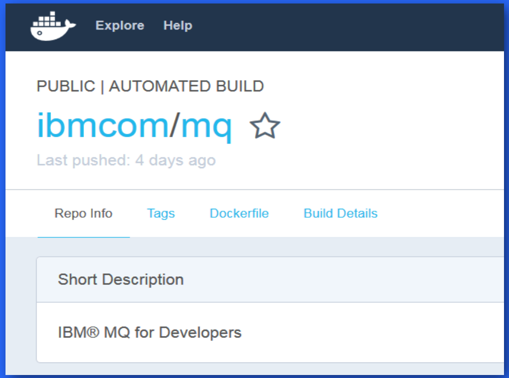
- Run it yourself in any cloud, public or private

- Let IBM host it for you with its new managed MQ service in IBM Cloud



- **MQ in Containers**

- **MQ has been supporting Docker containers since 2015 with images on Docker Hub and Docker Store and sample setups on Github**



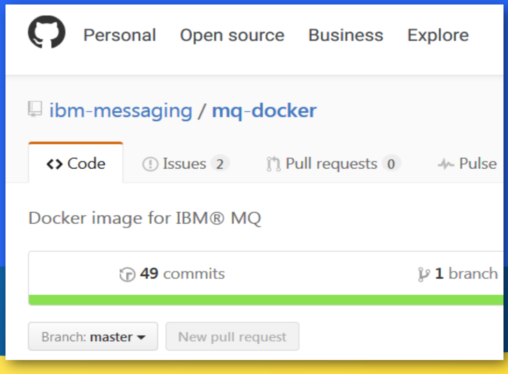
PUBLIC | AUTOMATED BUILD

ibmcom/mq

Last pushed: 4 days ago

Short Description

IBM® MQ for Developers

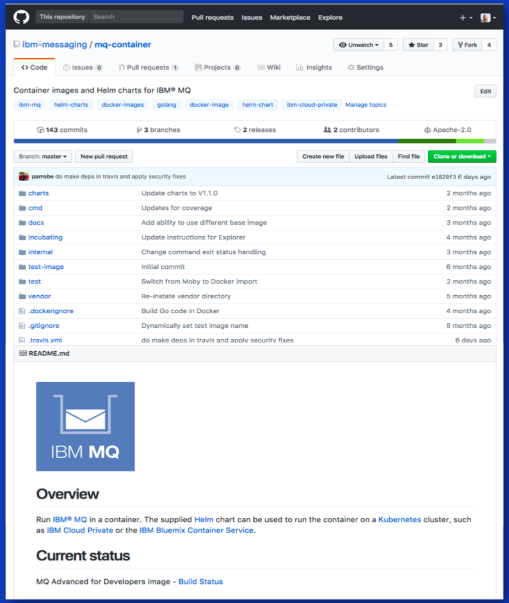


ibm-messaging / mq-docker

Docker image for IBM® MQ

49 commits | 1 branch

- **More recently it has been demonstrating how to get the most from containers using Kubernetes**



ibm-messaging / mq-container

Container images and Helm charts for IBM® MQ

143 commits | 3 branches | 2 releases | 2 contributors | Apache-2.0

parrobe do make deps in travis and apply security fixes — Latest commit e1829f3 6 days ago

charts	Update charts to V1.1.0	2 months ago
cmd	Updates for coverage	2 months ago
docs	Add ability to use different base image	3 months ago
incubating	Update instructions for Explorer	4 months ago
internal	Change command exit status handling	3 months ago
test-image	Initial commit	6 months ago
test	Switch from Moby to Docker import	2 months ago
vendor	Re-instate vendor directory	5 months ago
.dockerignore	Build Go code in Docker	4 months ago
.gitignore	Dynamically set test image name	5 months ago
.travis.yml	do make deps in travis and apply security fixes	6 days ago

README.md

IBM MQ

Overview

Run IBM® MQ in a container. The supplied Helm chart can be used to run the container on a Kubernetes cluster, such as IBM Cloud Private or the IBM Bluemix Container Service.

Current status

MQ Advanced for Developers image - Build Status

- **And now MQ Advanced is available as a fully supported product with IBM Cloud Private, a Kubernetes-based solution from IBM**

MQ on IBM Cloud

Provision queue managers directly into IBM Cloud

IBM owns the infrastructure and the responsibility to keep the systems up to date and running

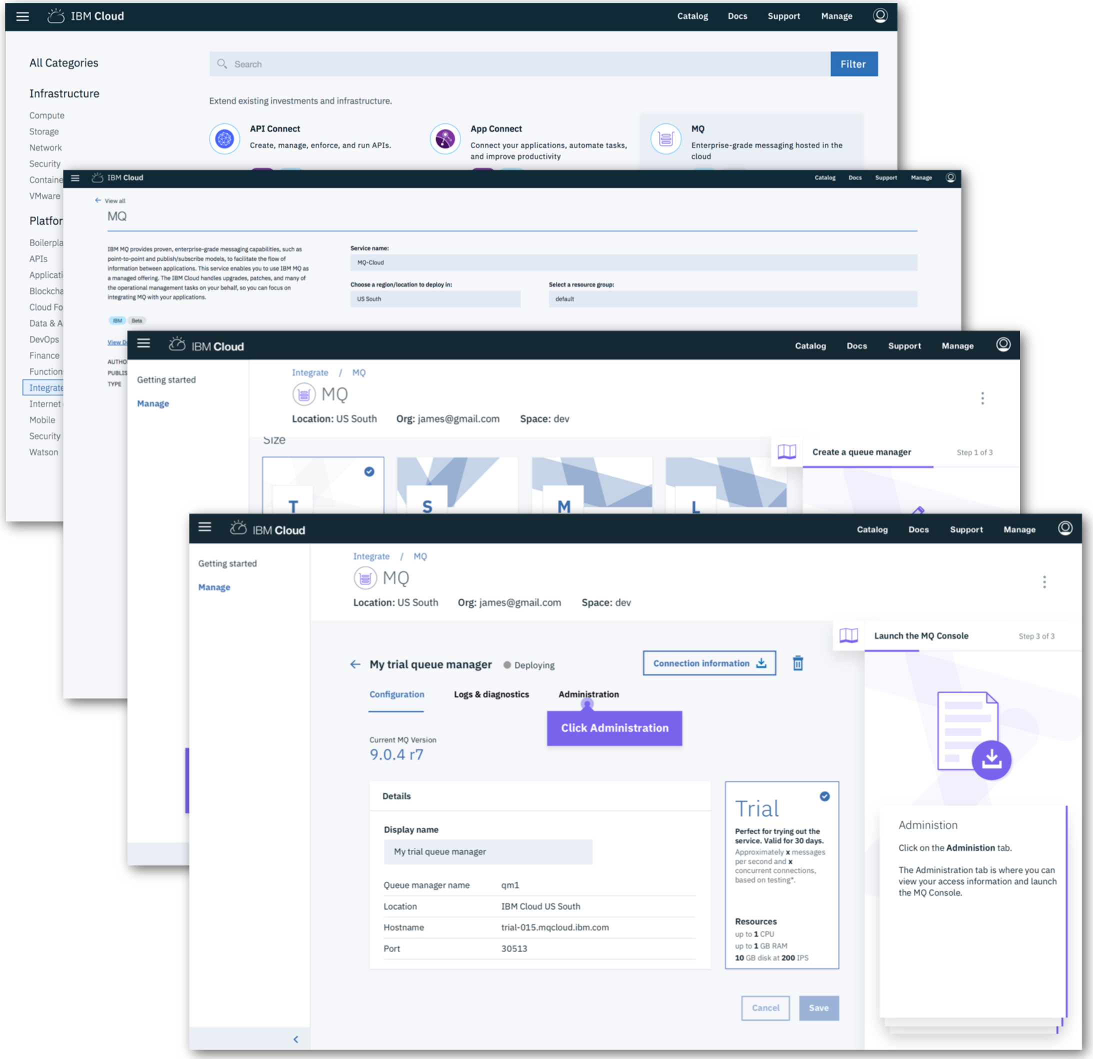
The customer owns the configuration and the monitoring of the messaging

Try the service for free at:

console.bluemix.net/catalog/services/mq

Hosted on

IBM Cloud



Extend existing investments and infrastructure.

API Connect Create, manage, enforce, and run APIs.

App Connect Connect your applications, automate tasks, and improve productivity

MQ Enterprise-grade messaging hosted in the cloud

MQ

IBM MQ provides proven, enterprise-grade messaging capabilities, such as point-to-point and publish/subscribe models, to facilitate the flow of information between applications. This service enables you to use IBM MQ as a managed offering. The IBM Cloud handles upgrades, patches, and many of the operational management tasks on your behalf, so you can focus on integrating MQ with your applications.

Service name: MQ-Cloud

Choose a region/location to deploy in: US South

Select a resource group: default

MQ

Location: US South **Org:** james@gmail.com **Space:** dev

My trial queue manager ● Deploying

Current MQ Version
9.0.4 r7

Details

Display name: My trial queue manager

Queue manager name	qm1
Location	IBM Cloud US South
Hostname	trial-015.mqcloud.ibm.com
Port	30513

Trial

Perfect for trying out the service. Valid for 30 days. Approximately **x** messages per second and **x** concurrent connections, based on testing*.

Resources
up to **1** CPU
up to **1** GB RAM
10 GB disk at **200** IPS

Administion

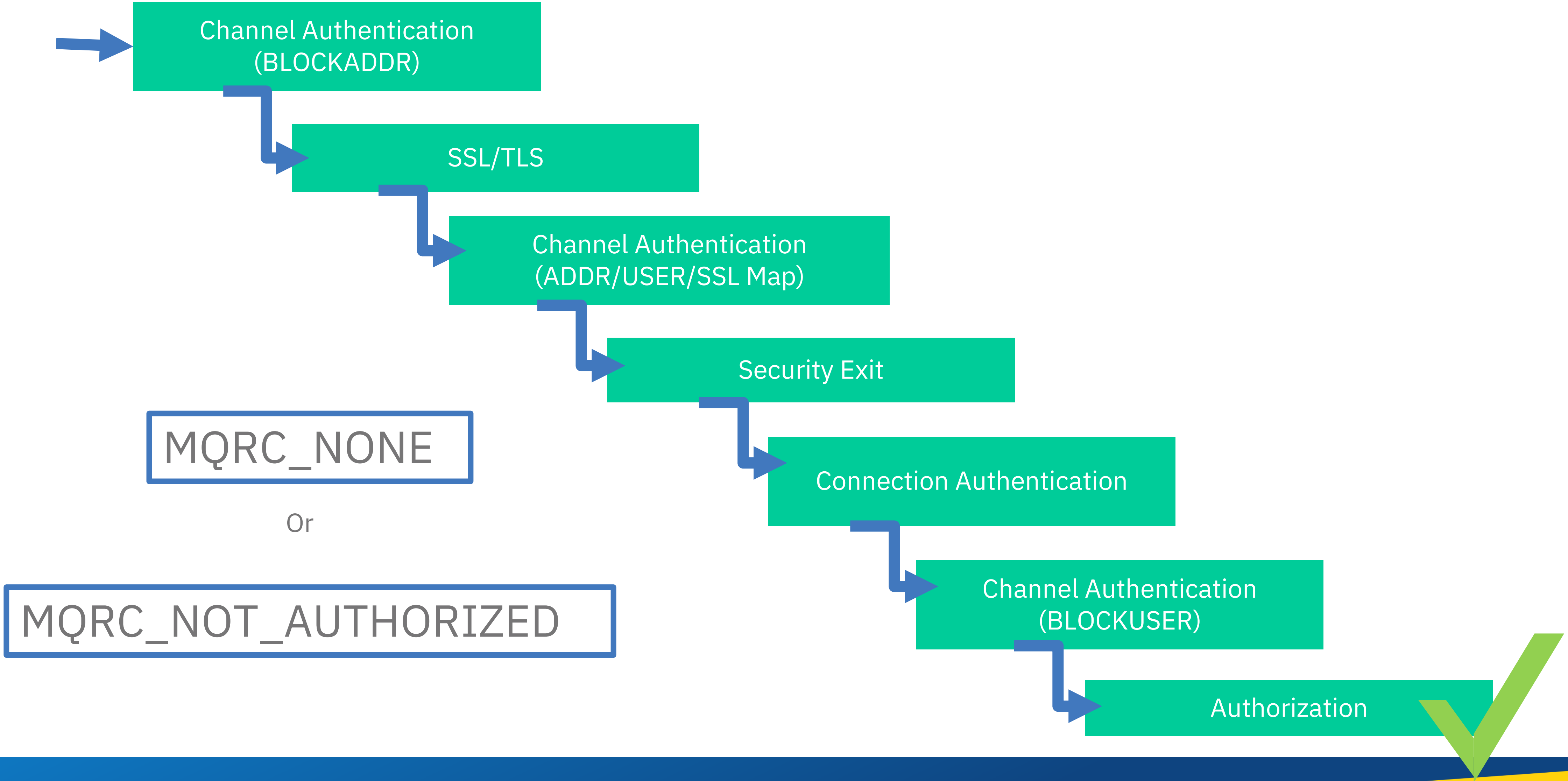
Click on the **Administion** tab.

The Administration tab is where you can view your access information and launch the MQ Console.

IBM MQ

Security

Security provided on Client to Queue Manager connections



Channel Authentication (BLOCKADDR)

SSL/TLS

Channel Authentication (ADDR/USER/SSL Map)

Security Exit

Connection Authentication

Channel Authentication (BLOCKUSER)

Authorization

MQRC_NONE

Or

MQRC_NOT_AUTHORIZED

IBM MQ

High Availibility

IBM MQ HA capabilities

Come to my other talks!
Benefitting from the MQ Appliance
Room: Zebrawood
Time: **Monday, 14:30PM** or
Wednesday 09:45AM

- Support for HA clusters and network storage
- Multi-instance queue managers (Windows, Linux, UNIX)
- IBM MQ Appliance

- Client connectivity
  - Automatic reconnection
  - CCDTs
  - Pre-connect exit

- Replicated Data Queue Managers (Linux)
- Queue-sharing groups (z/OS)
- Support for cloud orchestration frameworks e.g. Kubernetes, Docker Swarm, Apache Mesos

IBM **MQ**

IBM MQ

Administration

Administration and monitoring

Come to my other talks!

Administering MQ, The MQ Console and the MQ REST API!

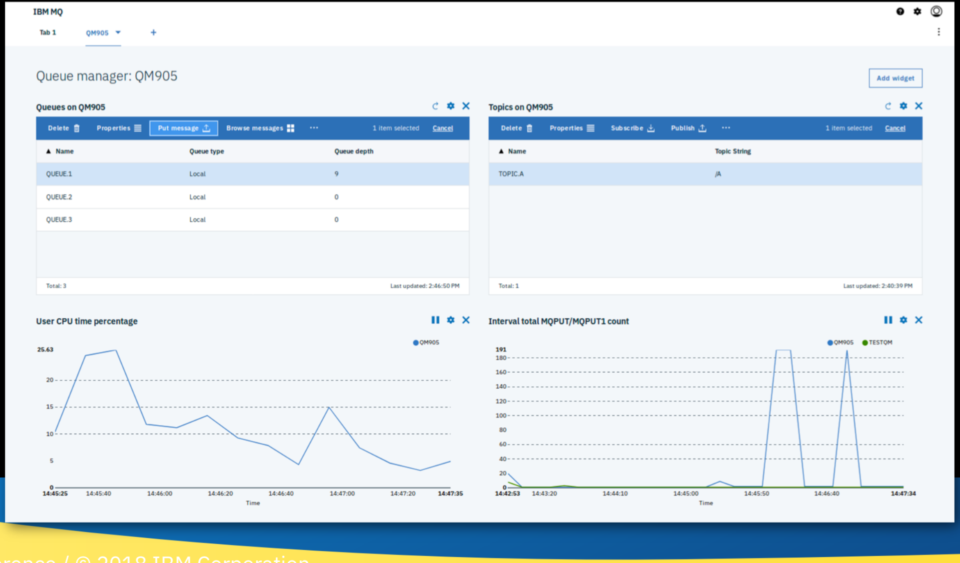
Room: Zebrawood

Time: **Tuesday, 8:30AM** or **Wednesday 13:00PM**

- Command line
- Scripting
- grammatic APIs
- ST API

- Tivoli and third-party tooling

- Web console



Queue manager: QM905

Queues on QM905

▲ Name	Queue type	Queue depth
QUEUE.1	Local	9
QUEUE.2	Local	0
QUEUE.3	Local	0

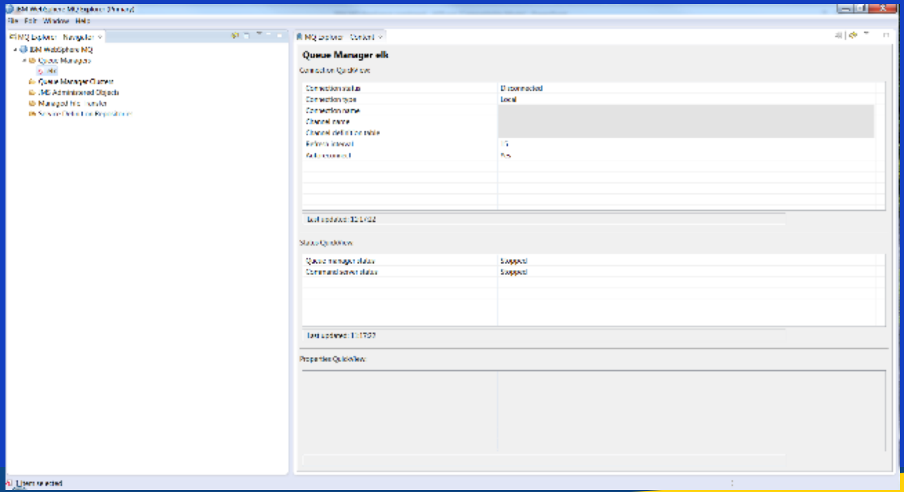
Total: 3 Last updated: 2:46:50 PM

Topics on QM905

▲ Name	Topic String
TOPIC.A	/A

Total: 1 Last updated: 2:40:39 PM

- GUI tooling



IBM MQ

MQ Advanced

MQ Advanced Message Security

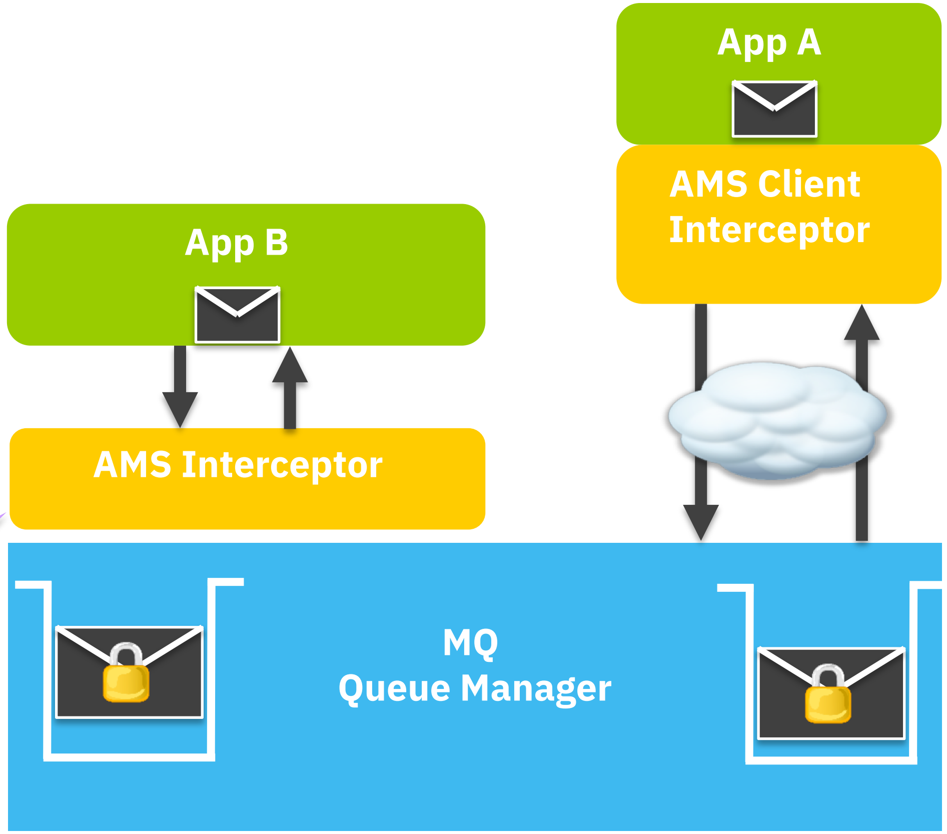
- **Secures application data even before it is passed to MQ**
- **Upgrade from base MQ**
  - No changes to existing applications or network required

MQ standard security:

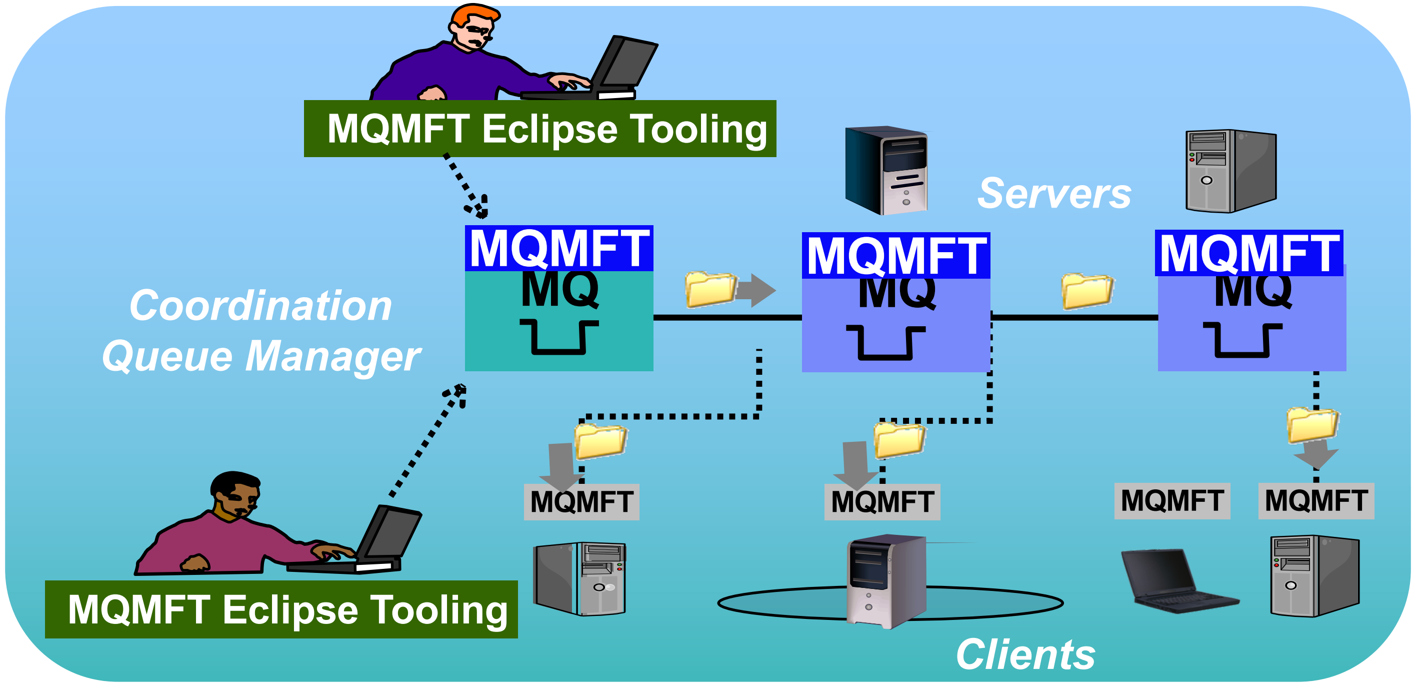
- Industry standard TLS channels (256-bit)
- Certified for Common Criteria
- Authentication is based on Operating System identifier of local process
- Message data can be encrypted in transport but not when it resides in the queues

MQ Advanced Message Security adds:

- Authentication policies are based on certificates associated with each application
- Message data is protected end-to-end – including when it resides in queues
- Much finer granularity in security policies
- No changes needed to applications or queues



MQ Managed File Transfer



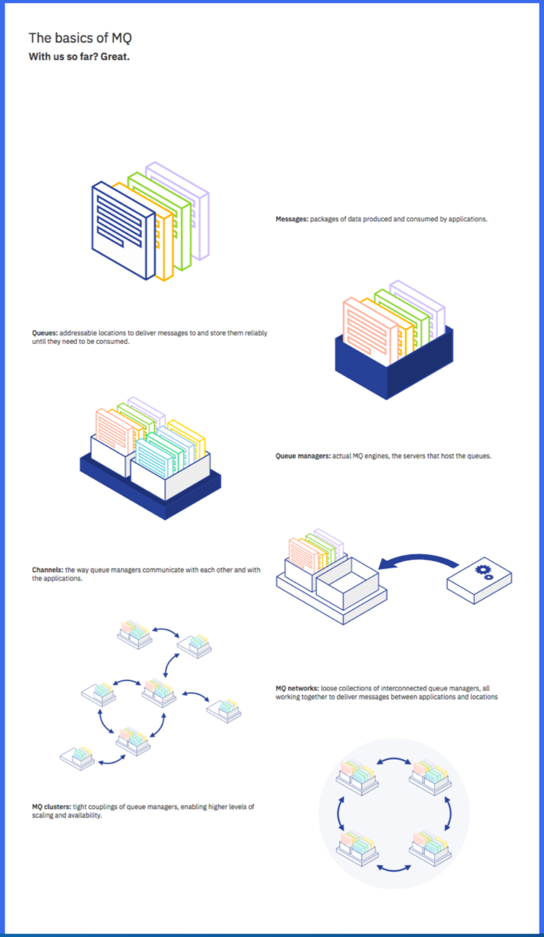
Where to go now?

- LearnMQ

- Finding it hard to get developers started with MQ?

- Point them to: developer.ibm.com/messaging/learn-mq

- Totally new to MQ? Learn the basics



The basics of MQ

With us so far? Great.

Messages: packages of data produced and consumed by applications.

Queues: addressable locations to deliver messages to and store them reliably until they need to be consumed.

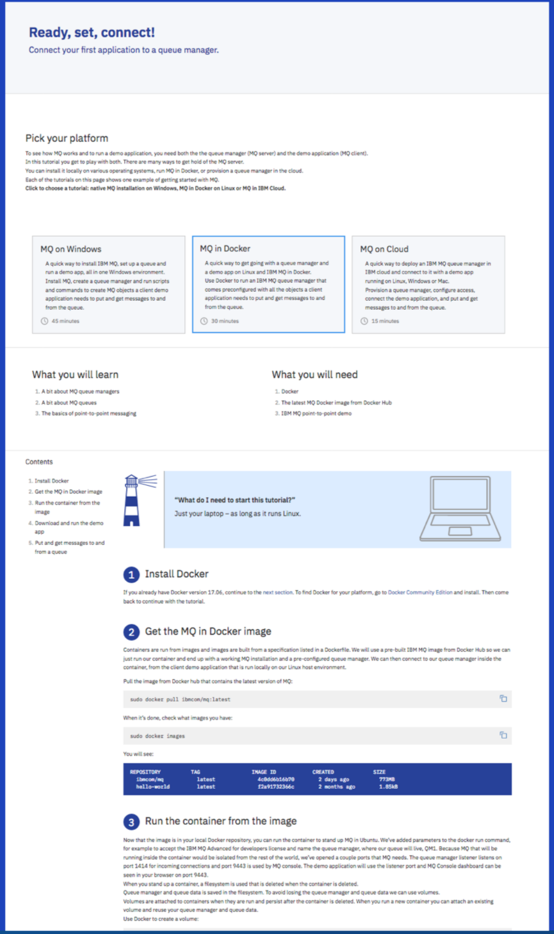
Queue managers: actual MQ engines, the servers that host the queues.

Channels: the way queue managers communicate with each other and with the applications.

MQ networks: loose collections of interconnected queue managers, all working together to deliver messages between applications and locations

MQ clusters: tight couplings of queue managers, enabling higher levels of scaling and availability.

- Step-by-step guide to getting up and running with MQ



Ready, set, connect!

Connect your first application to a queue manager.

Pick your platform

To see how MQ works and to run a demo application, you need both the the queue manager (MQ server) and the demo application (MQ client).
In this tutorial you get to play with both. There are many ways to get hold of the MQ server.
You can install it locally on various operating systems, run MQ in Docker, or provision a queue manager in the cloud.
Each of the tutorials on this page shows one example of getting started with MQ.
Click to choose a tutorial: native MQ installation on Windows, MQ in Docker on Linux or MQ in IBM Cloud.

MQ on Windows
A quick way to install IBM MQ, set up a queue and run a demo app, all in one Windows environment. Install MQ, create a queue manager and run scripts and commands to create MQ objects a client demo application needs to put and get messages to and from the queue.
45 minutes

MQ in Docker
A quick way to get going with a queue manager and a demo app on Linux and IBM MQ in Docker.
Use Docker to run an IBM MQ queue manager that comes preconfigured with all the objects a client application needs to put and get messages to and from the queue.
30 minutes

MQ on Cloud
A quick way to deploy an IBM MQ queue manager in IBM cloud and connect to it with a demo app running on Linux, Windows or Mac.
Provision a queue manager, configure access, connect the demo application, and put and get messages to and from the queue.
15 minutes

What you will learn

1. A bit about MQ queue managers
2. A bit about MQ queues
3. The basics of point-to-point messaging

What you will need

1. Docker
2. The latest MQ Docker image from Docker Hub
3. IBM MQ point-to-point demo

Contents

1. Install Docker
2. Get the MQ in Docker image
3. Run the container from the image
4. Download and run the demo app
5. Put and get messages to and from a queue

"What do I need to start this tutorial?"
Just your laptop – as long as it runs Linux.

1 Install Docker

If you already have Docker version 17.06, continue to the next section. To find Docker for your platform, go to Docker Community Edition and install. Then come back to continue with the tutorial.

2 Get the MQ in Docker image

Containers are run from images and images are built from a specification listed in a Dockerfile. We will use a pre-built IBM MQ image from Docker Hub so we can just run our container and end up with a working MQ installation and a pre-configured queue manager. We can then connect to our queue manager inside the container, from the client demo application that is run locally on our Linux host environment.

Pull the image from Docker hub that contains the latest version of MQ:

```
sudo docker pull ibmcom/mq:latest
```

When it's done, check what images you have:

```
sudo docker images
```

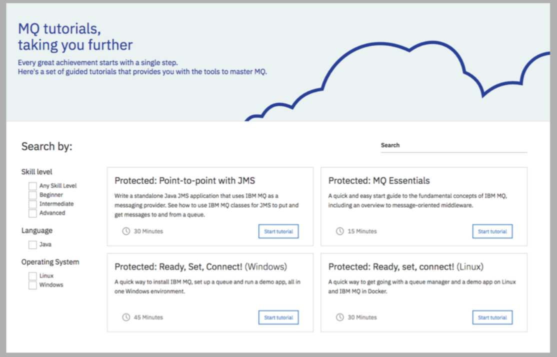
You will see:

```
REPOSITORY       TAG          IMAGE ID          CREATED          SIZE
  ibmcom/mq        latest       4c0dd6b16b70      2 days ago        773MB
  hello-world      latest       f2a91732366c      2 months ago      1.85kB
```

3 Run the container from the image

Now that the image is in your local Docker repository, you can run the container to stand up MQ in Ubuntu. We've added parameters to the docker run command, for example to accept the IBM MQ Advanced for developers license and name the queue manager, where our queue will live, QM1. Because MQ that will be running inside the container would be isolated from the rest of the world, we've opened a couple ports that MQ needs. The queue manager listener listens on port 1414 for incoming connections and port 9443 is used by MQ console. The demo application will use the listener port and MQ Console dashboard can be seen in your browser on port 9443.
When you stand up a container, a filesystem is used that is deleted when the container is deleted.
Queue manager and queue data is saved in the filesystem. To avoid losing the queue manager and queue data we can use volumes.
Volumes are attached to containers when they are run and persist after the container is deleted. When you run a new container you can attach an existing volume and reuse your queue manager and queue data.
Use Docker to create a volume:

- Tutorials on building your applications



MQ tutorials,
taking you further

Every great achievement starts with a single step.
Here's a set of guided tutorials that provides you with the tools to master MQ.

Search by:

Search

Skill level
□ Any Skill Level
□ Beginner
□ Intermediate
□ Advanced

Language
□ Java

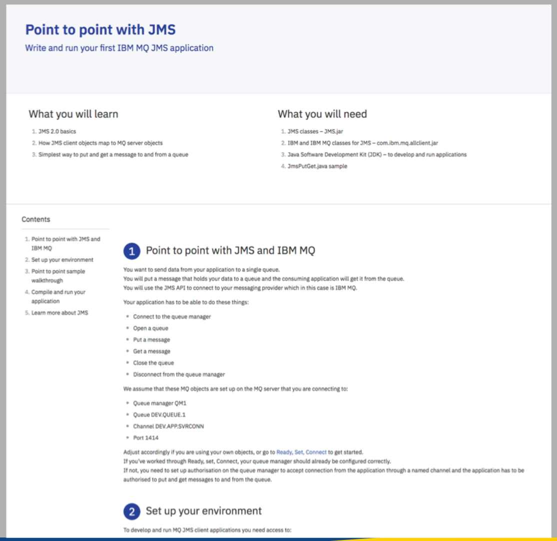
Operating System
□ Linux
□ Windows

Protected: Point-to-point with JMS
Write a standalone Java JMS application that uses IBM MQ as a messaging provider. See how to use IBM MQ classes for JMS to put and get messages to and from a queue.
30 Minutes
Start tutorial

Protected: MQ Essentials
A quick and easy start guide to the fundamental concepts of IBM MQ, including an overview to message-oriented middleware.
15 Minutes
Start tutorial

Protected: Ready, Set, Connect! (Windows)
A quick way to install IBM MQ, set up a queue and run a demo app, all in one Windows environment.
45 Minutes
Start tutorial

Protected: Ready, set, connect! (Linux)
A quick way to get going with a queue manager and a demo app on Linux and IBM MQ in Docker.
30 Minutes
Start tutorial



Point to point with JMS

Write and run your first IBM MQ JMS application

What you will learn

1. JMS 2.0 basics
2. How JMS client objects map to MQ server objects
3. Simplest way to put and get a message to and from a queue

What you will need

1. JMS classes – JMS.jar
2. IBM and IBM MQ classes for JMS – com.ibm.mq.allclient.jar
3. Java Software Development Kit (JDK) – to develop and run applications
4. JmsPutGet.java sample

Contents

1. Point to point with JMS and IBM MQ
2. Set up your environment
3. Point to point sample walkthrough
4. Compile and run your application
5. Learn more about JMS

1 Point to point with JMS and IBM MQ

You want to send data from your application to a single queue.
You will put a message that holds your data to a queue and the consuming application will get it from the queue.
You will use the JMS API to connect to your messaging provider which in this case is IBM MQ.

Your application has to be able to do these things:

- Connect to the queue manager
- Open a queue
- Put a message
- Get a message
- Close the queue
- Disconnect from the queue manager

We assume that these MQ objects are set up on the MQ server that you are connecting to:

- Queue manager QM1
- Queue DEV.QUEUE.1
- Channel DEV.APP.SVRCONN
- Port 1414

Adjust accordingly if you are using your own objects, or go to Ready, Set, Connect to get started.
If you've worked through Ready, set, Connect, your queue manager should already be configured correctly.
If not, you need to set up authorisation on the queue manager to accept connection from the application through a named channel and the application has to be authorised to put and get messages to and from the queue.

2 Set up your environment

To develop and run MQ JMS client applications you need access to:

Where do I get more information?

IBM MQ Knowledge Center
http://www.ibm.com/software/integration/wmq/library/

IBM Messaging developerWorks
developer.ibm.com/messaging

Youtube
https://www.youtube.com/user/IBMmessagingMedia

Copyright and Trademarks

Thanks for listening

Questions?

Sam Goulden IBM UK, IBM MQ L3 Service

sgoulde4@uk.ibm.com