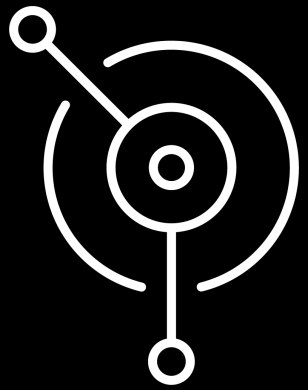
# MQ Application Development Essentials & Anti-Patterns

Jamie Squibb
IBM UK

November 2019
Session JM

GSE UK Conference 2019



# IBM **MQ**

Application Essentials: Introduction

# Developer engagement

Developers can find it hard to get started with MQ

How do we make it easier for new application teams to pick up MQ and integrate with their development practices?

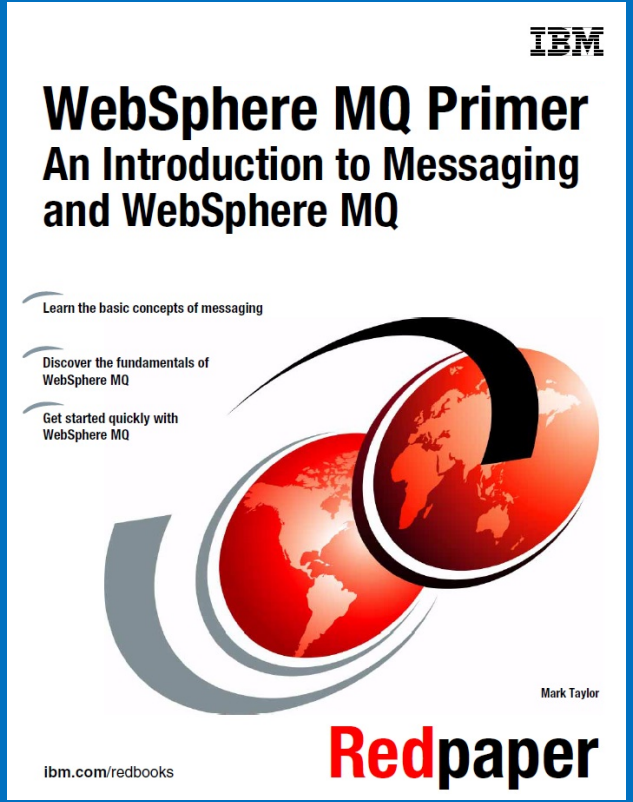
Not always been a product focus – spent more time on administration enablement

**Simplifying application development and support**

- Improves developer experience
- Indirectly improves the lives of administrators
- Better MQ applications = fewer headaches for administrators

# Developer engagement

This is what developer engagement used to look like



# Mission statement

To enable a user instructed to use MQ for the first time, to go from zero understanding to ***running*** a sample application in a sandbox environment with a fundamental understanding of MQ concepts in 2 hours

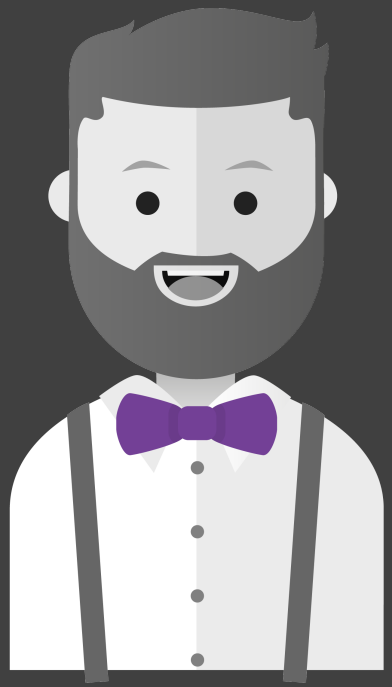
To enable an application developer, instructed to use MQ for the first time, to go from zero understanding to ***writing*** their first MQ application in the language and environment of their choice within an afternoon

**developer**: so what?

Just show me an API and I'll code.

Working code speaks for itself.

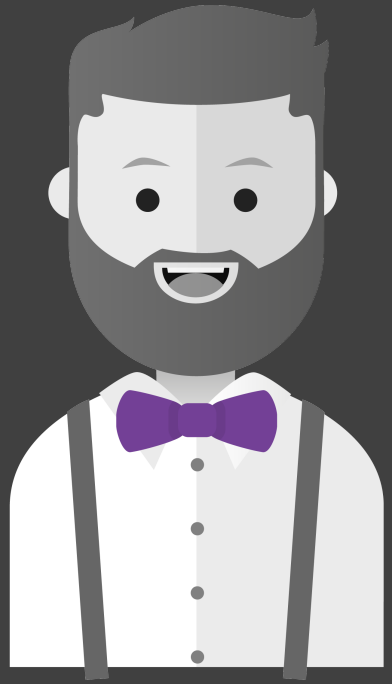
Ready to learn and excited to grow marketable skills.



# **developer**: so what?

Last year, MQ celebrated its 25th birthday and so is Andre, a new application developer.

He needs to deliver a reliable MQ application for his business.



# *Andre*

Application Developer

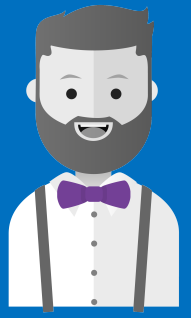
I need to deliver a new application

My job is to get code working

**Skills**

Java | C | Ruby | REST | Linux

# **developer**: insight



## responsibilities

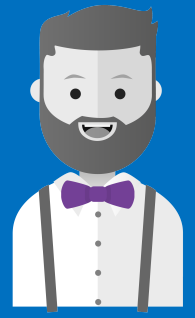
- App design & logic
- Getting new app connected to the MQ Network
- DevOps
- Incident resolution
- Getting to done!

## needs

- Find information on MQ
- Core concepts of MQ
- Build applications with MQ
- Some community support
- Marketable skills

## likes

- Samples, assets and code to get started (GitHub)
- Working in his preferred dev environment
- Doing stuff the *right way*
- Minimal setup
- Enjoying  work
- Help when he needs it

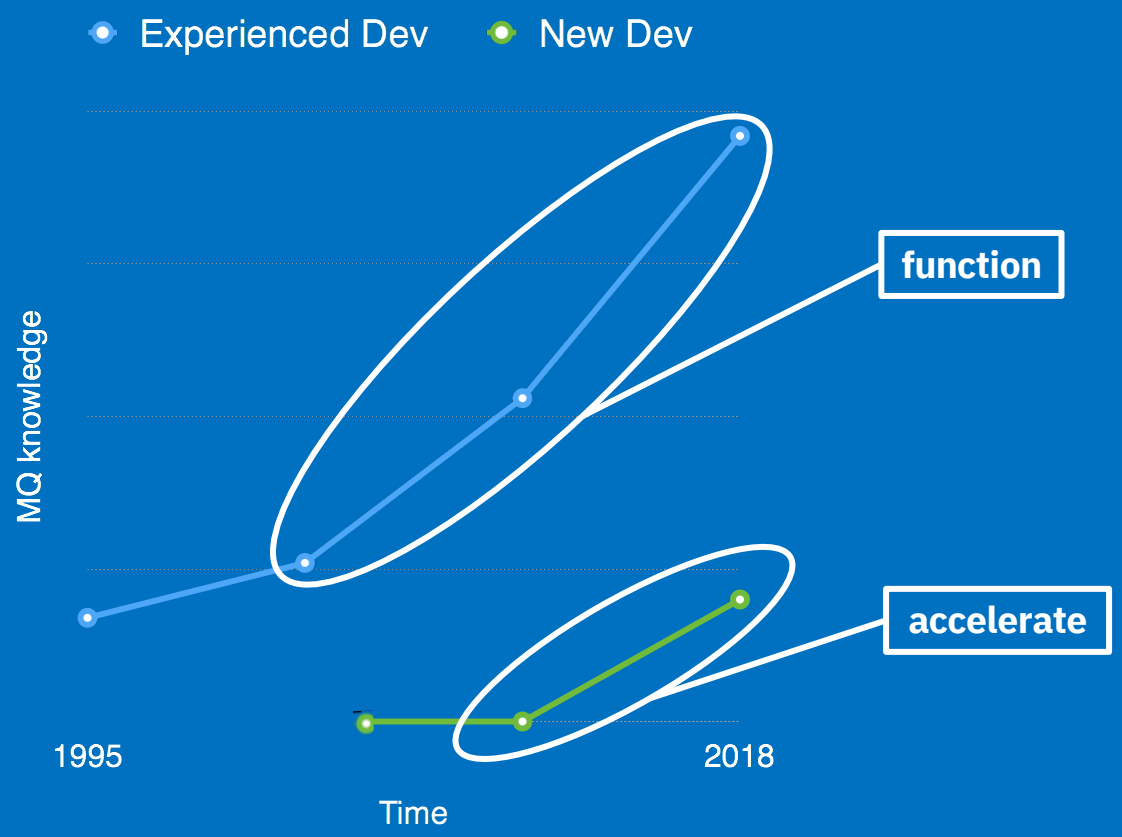


Just one thing...

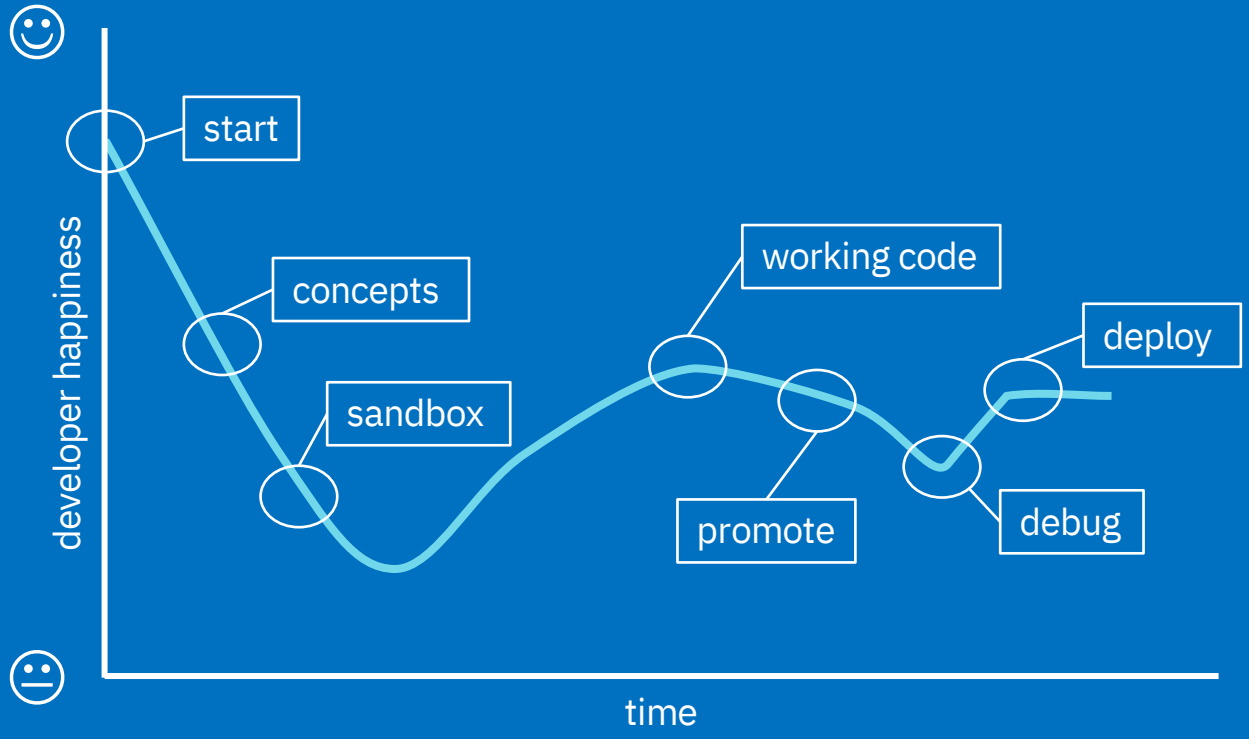
# He’s not used IBM MQ before

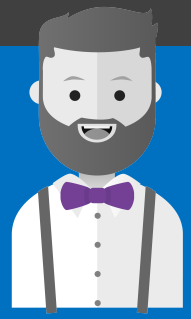
 

# **developer**: knowledge cycle



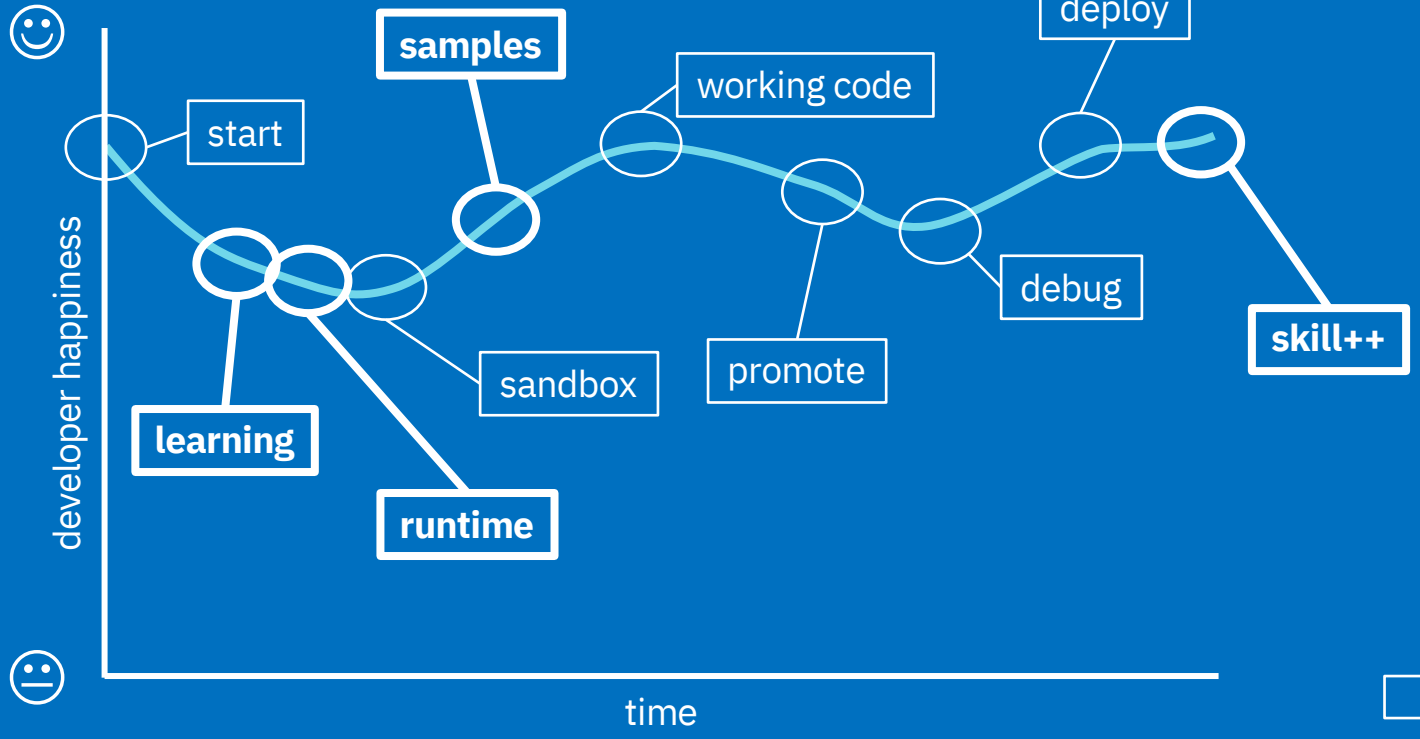
# **dev experience**: early research

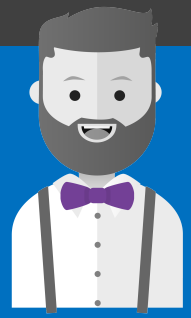


= stages of app development

# **dev experience**: positive direction

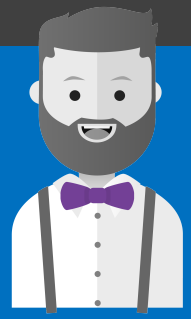


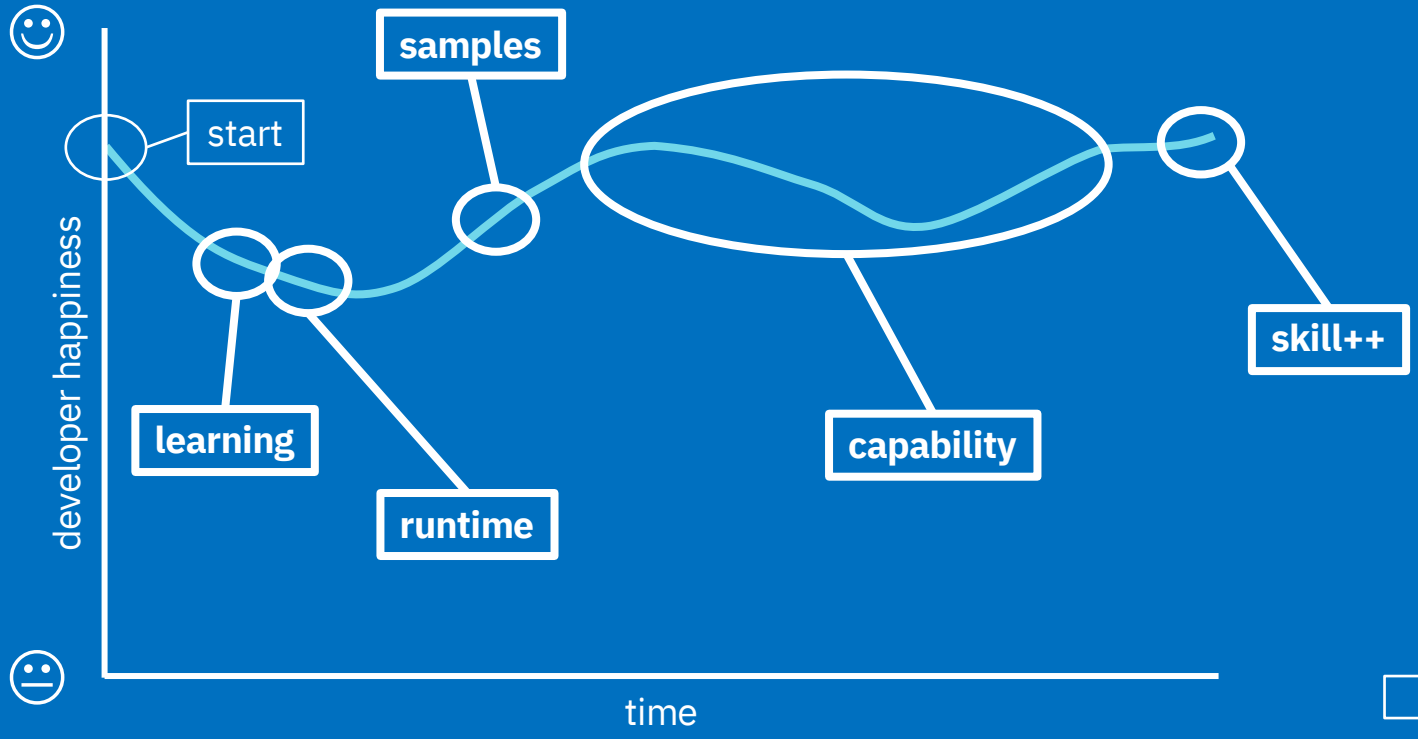



= stages of app development

= **our focus areas**

# **dev experience**: positive direction






= stages of app development

= **our focus areas**

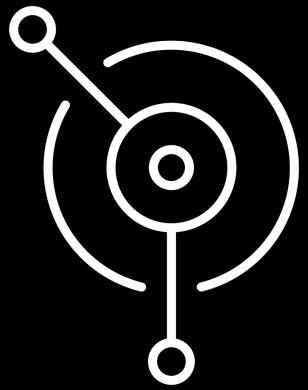
# **dev experience**: deployment landscape

More than just APIs: scale, reliability, security and extensibility capabilities are critical.

Need working production ready code to succeed.

IBM MQ provides enterprise grade messaging and is a very marketable skill.

GSE UK Conference 2019



# IBM **MQ**

## Application Essentials: Getting Started

## Welcome to messaging, made easy.

IBM MQ is the world's most powerful messaging solution.
LearnMQ will teach you how to master it.

# ibm.biz/learn-mq

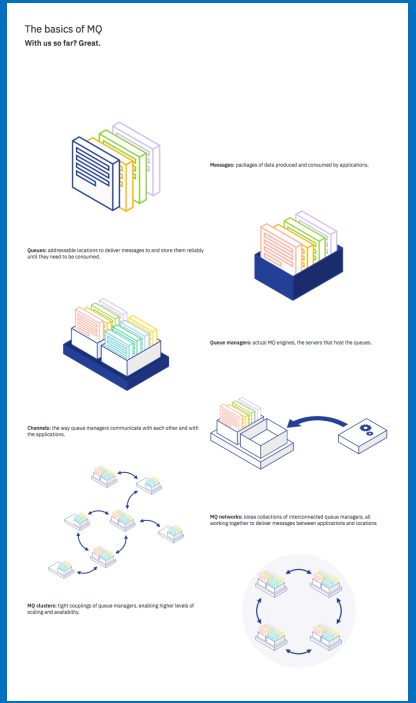
# LearnMQ

Finding it hard to get developers started with MQ?

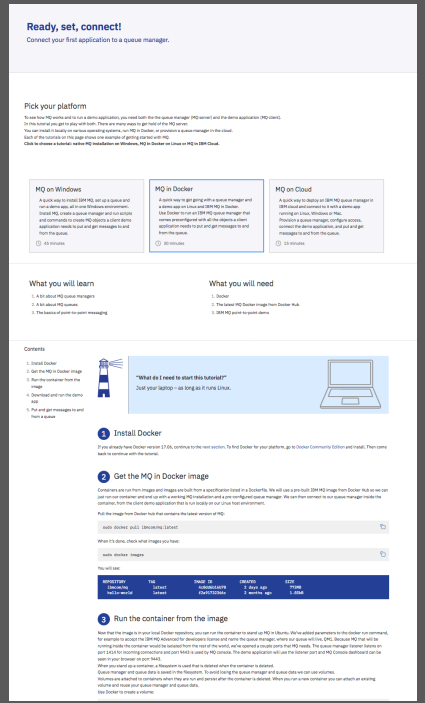
Point them to:

**developer.ibm.com/messaging/learn-mq**

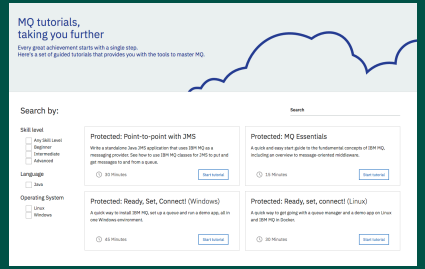
## Totally new to MQ? Learn the basics

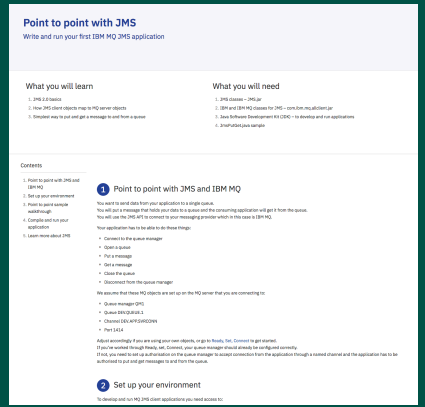



## Step-by-step guides to getting up and running with MQ




## Tutorials on building your applications and avoiding bad practices



# 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.

MQ Essentials (5-10 mins)

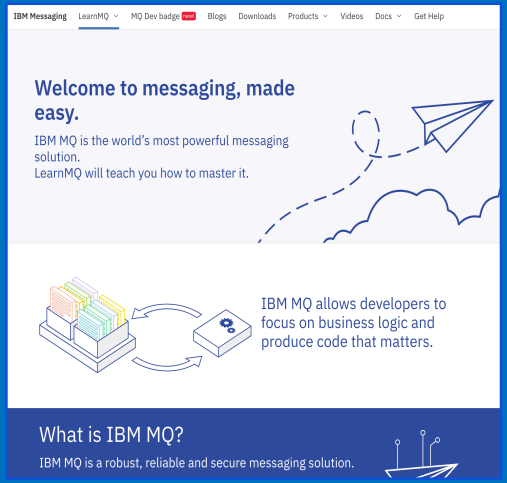
Get a queue manager (10-15 mins)

Hello world (10-15 mins)

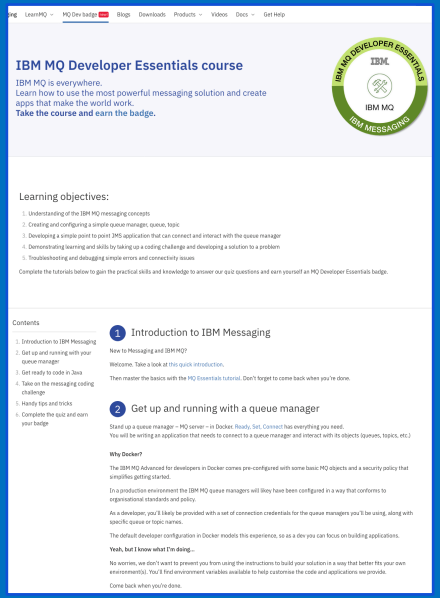
25 years in 25 minutes!

# LearnMQ site

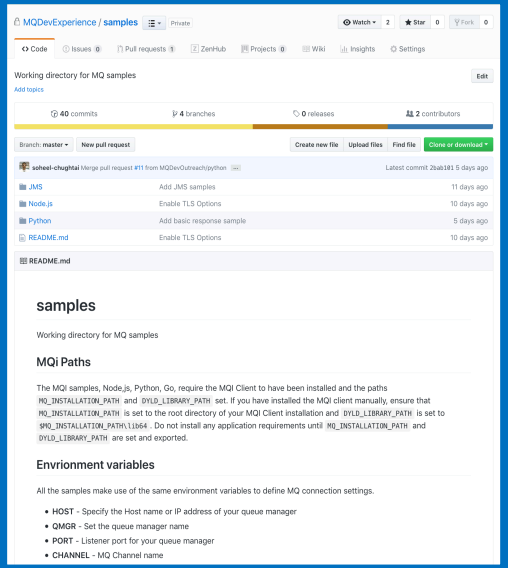
## LearnMQ




## Developer Essentials Badge and challenge



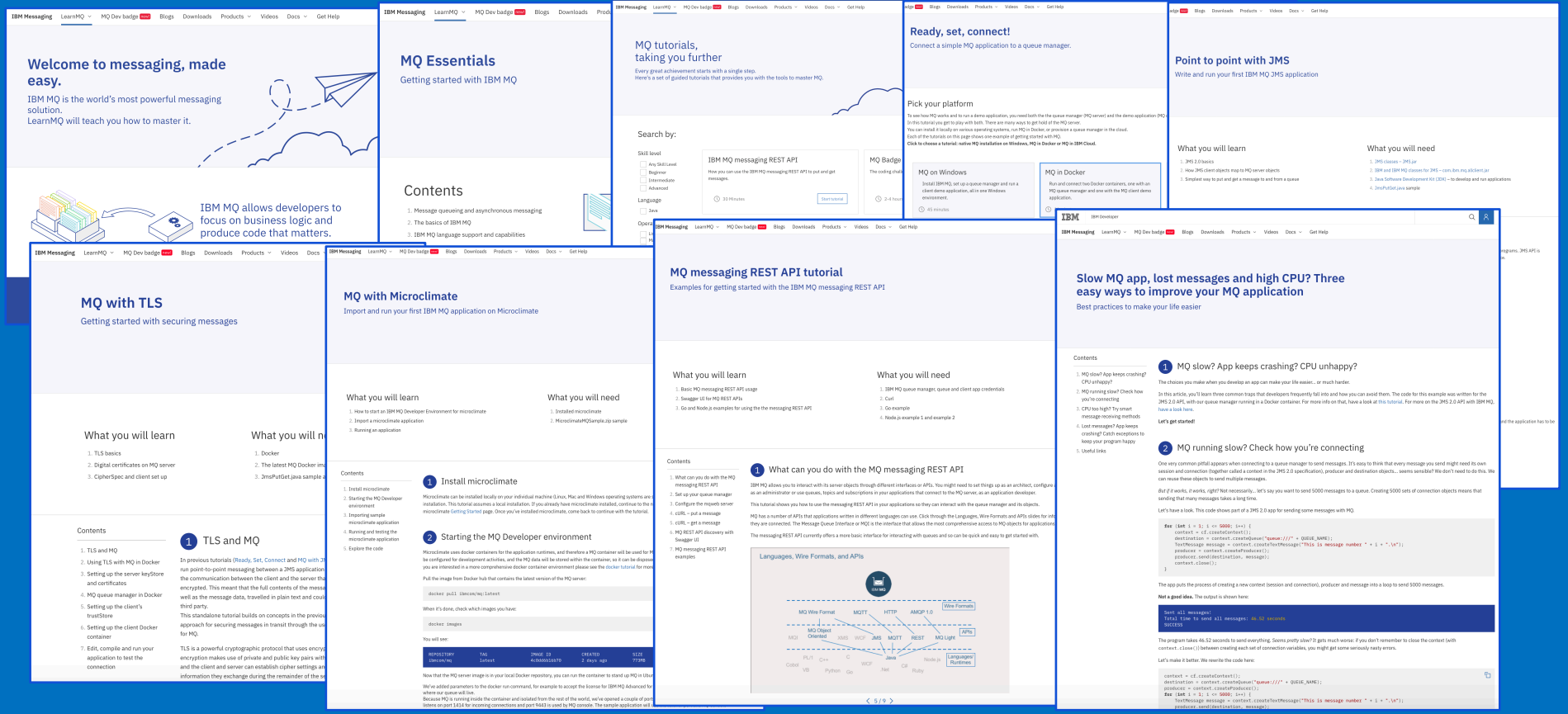

## Samples and patterns



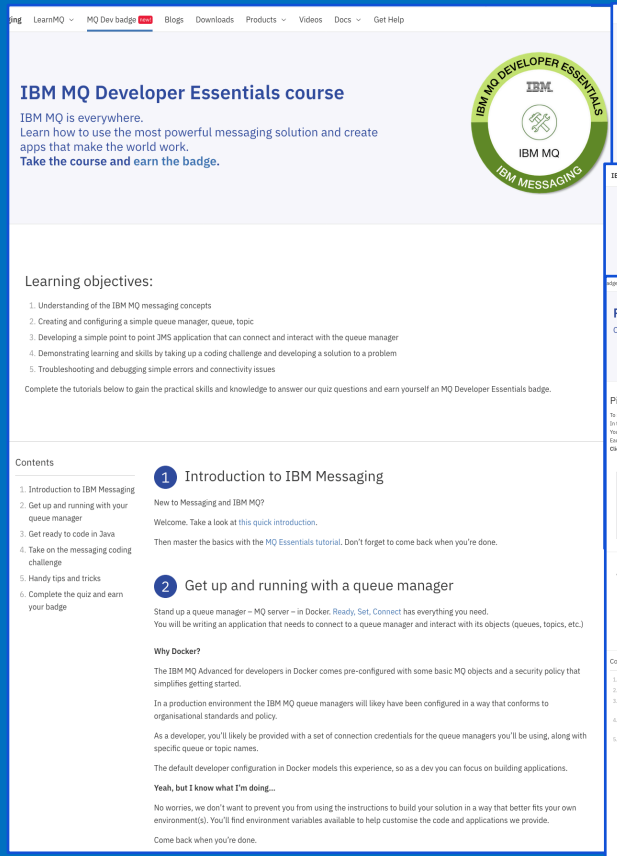

http://ibm.biz/mq-dev-patterns

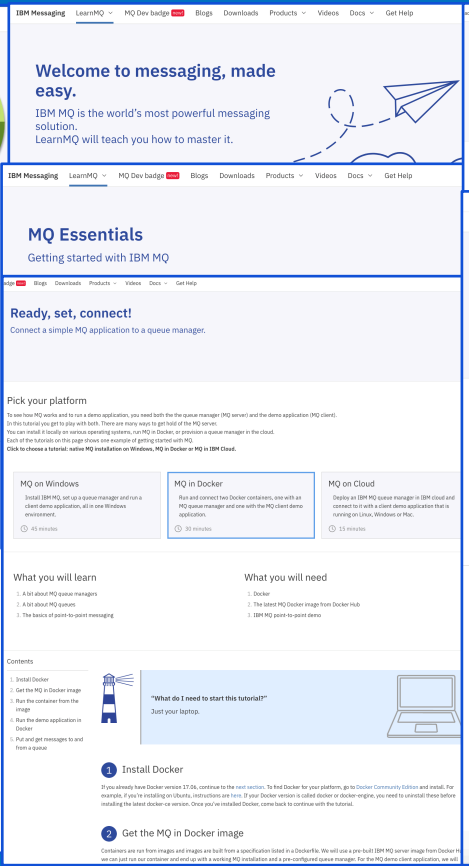
(No longer need to install the product just to see sample code!)

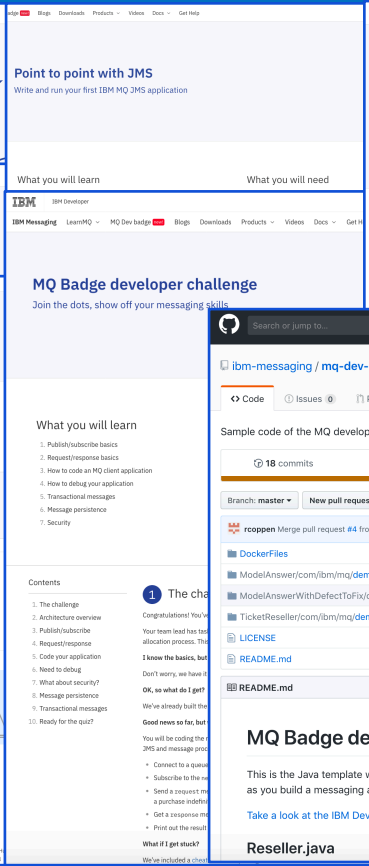
# LearnMQ tutorials

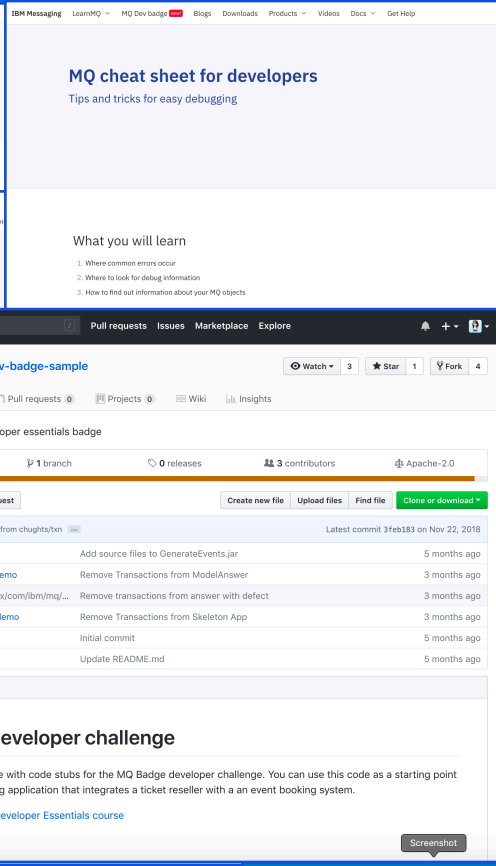


# MQ Developer Essentials Badge

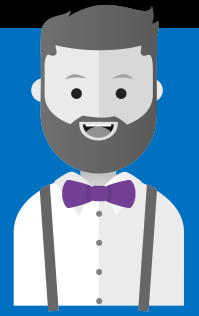


# Why a Badge?



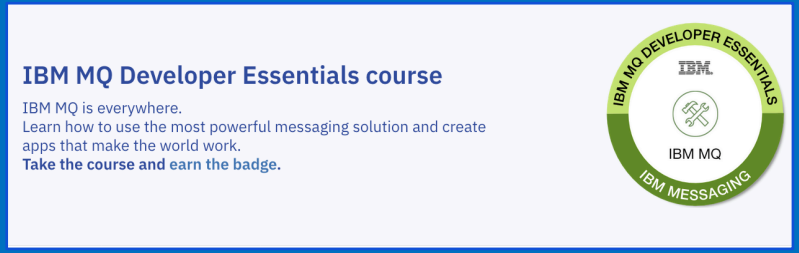
Sets a modest but achievable bar for a new developer to attain in a short period of time

Guarantees a consistent set of terms and concepts for all new MQ devs

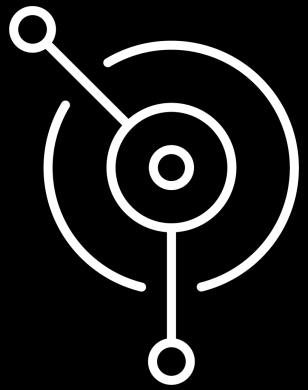
Doesn't require weeks to achieve

Gives MQ administrators:

- a single place to point new developers at
- a known starting point once developers have completed it



GSE UK Conference 2019



# IBM **MQ**

## Application Essentials: Recent Updates

# Demonstrating the simplicity of MQ

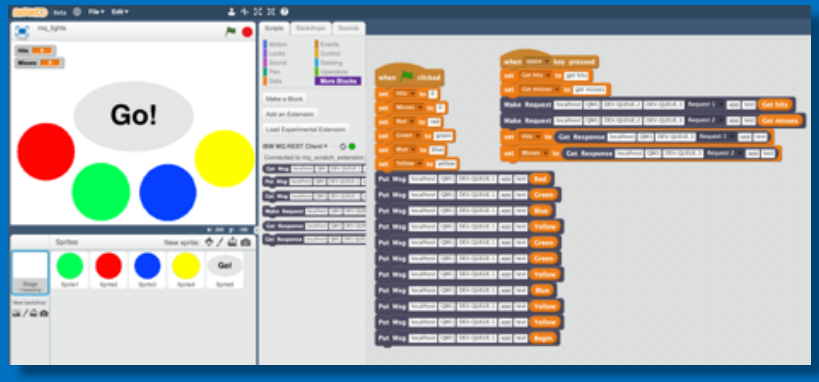
There's nothing like flashing lights and wires to grab people's attention. We want everyone to know how easy it is to write messaging applications and how powerful MQ is in supporting them

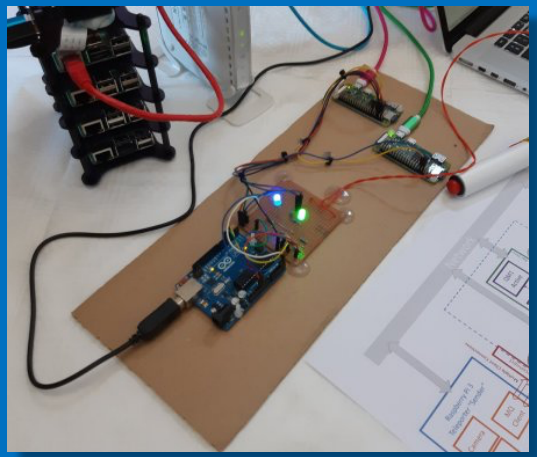
Ever tried **Scratch,** a graphical way to code, aimed at kids but ideal to show how easily asynchronous messaging can improve your applications with an MQ plugin

**ibm.biz/ibmmq-scratch**

Heard of the **Raspberry PI**? You think MQ is a heavyweight solution? We run an HA pair of queue managers on two $5 Raspberry PI Zeros!

**ibm.biz/ibmmq-pi**



# Multiple APIs and protocols

IBM MQ supports multiple APIs and multiple client protocols. Both proprietary and open.

APIs: MQI, JMS, MQ Light, REST ...

Protocols: MQ, AMQP, MQTT, HTTP

These support a wide range of application styles, from the simplest of messaging needs through to the richest

MQ is the transport; messages produced from any API or protocol can be received by any other API or protocol


## APIs

**MQI**
Exposes the full set of MQ capabilities
Uses the MQ protocol

**JMS 2.0**
Supports the full JMS API for use in many JSE or JEE environments
Uses the MQ protocol

**MQ Light**
A simple pub/sub messaging API
Uses the AMQP 1.0 protocol

## Protocols

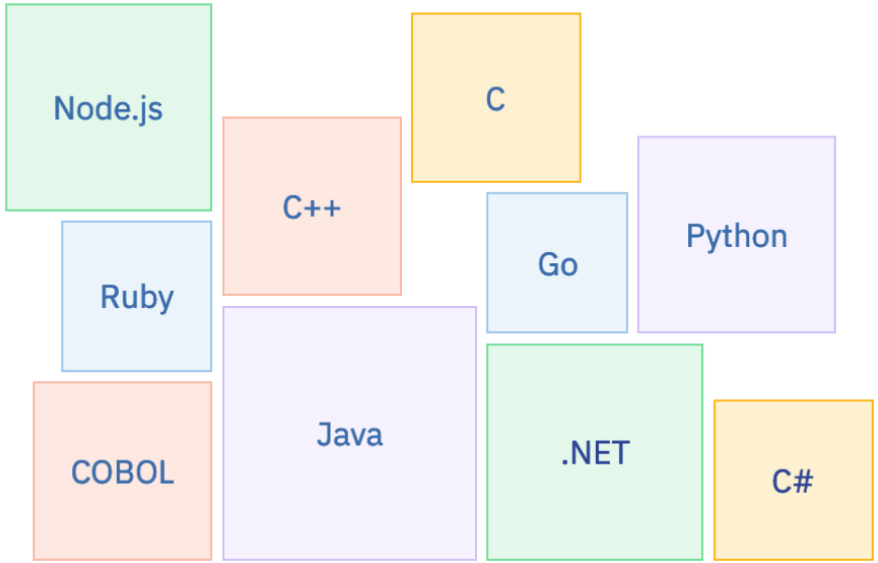
**MQ**
MQ's highly reliable and performant messaging protocol

**MQTT**
Supports the open standard MQTT API and protocol
Open source Eclipse Paho clients

**AMQP**
Support for AMQP 1.0 enables support for open source clients such as Qpid Proton

**REST**
A very simple but secure messaging API over REST
...

# MQ speaks your language




Recent updates

| | | |
|---|---|---|
| APIs & Protocols | REST | Go |
| Node.js | Maven | MQLight |
| Spring | Docker | LTS & CD |

# Developing applications

Build your applications simply, with no need for an MQ installation

- Pull Java directly from the Maven repository since MQ 9.0.4 CD
- MQ 9.1.1 CD added the **SDK** to the MQ redistributable client
- The redistributable client is now available directly, no need to log into IBM

**ibm.biz/mqclientdownload**


Develop your applications on the platform of your choice with the addition of the MacOS version of the 9.1.1 MQ client and SDK for Developers

**ibm.biz/mqmacos**

*(The MQ for MacOS toolkit includes runmqsc)*

# New MQI languages

The MQI API exposes the richest MQ messaging capabilities

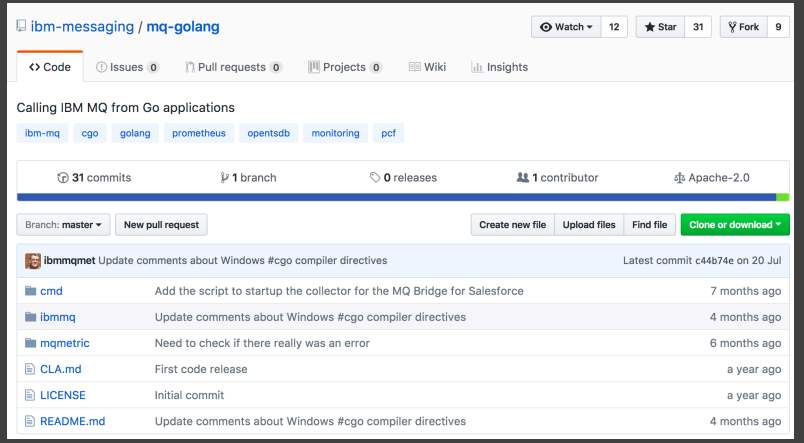
Traditionally mostly used by C and COBOL

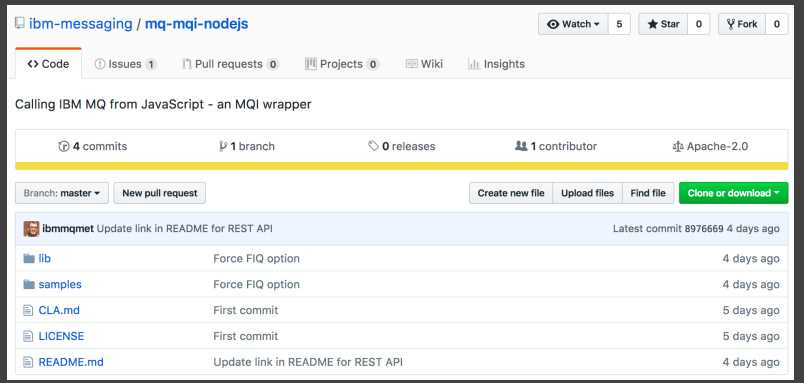
Other bindings have been available

New interfaces available for GO and Node.js designed around full MQI function

Provided as as-is open source, let us know if there's a need to take these further

Built on the C MQI library



# New MQI languages - examples

Intent to make it easy to write for MQ from newer languages and environments

In a natural style for the language

e.g. strings, not padded-fixed-lengths

Helps you to develop apps wherever you need

For **Node.js JavaScript** see **github.com/ibm-messaging/mq-mqi-nodejs**

or **require('ibmmq')** from NPM

For **Go** bindings see **github.com/ibm-messaging/mq-golang**

Both include sample programs to help get started

```
mqmd = new mq.MQMD();
pmo = new mq.MQPMO();
msg = "Hello from Node";

mq.Put(hObj,mqmd,pmo,msg,function(err) {
  if (err) {
    console.log(err);
  }
});
```

```
mqmd := ibmmq.NewMQMD()
pmo  := ibmmq.NewMQPMO()
mqmd.Format = "MQSTR"
msg  := "Hello from Go"

buffer := []byte(msg)
err = qObject.Put(mqmd, pmo, buffer)

if err != nil {
  fmt.Println(err)
}
```

# Availability of runtimes

Docker container with full MQ server function

- Creates sample objects for developers
- Queues, topics, userids etc

See https://hub.docker.com/r/ibmcom/mq/

Redistributable Client packages now easier to access

- To make it easy for developers to package standalone applications
- Perhaps in a container

public.dhe.ibm.com/ibmdl/export/pub/software/websphere/messaging/mqdev/redist

# REST messaging

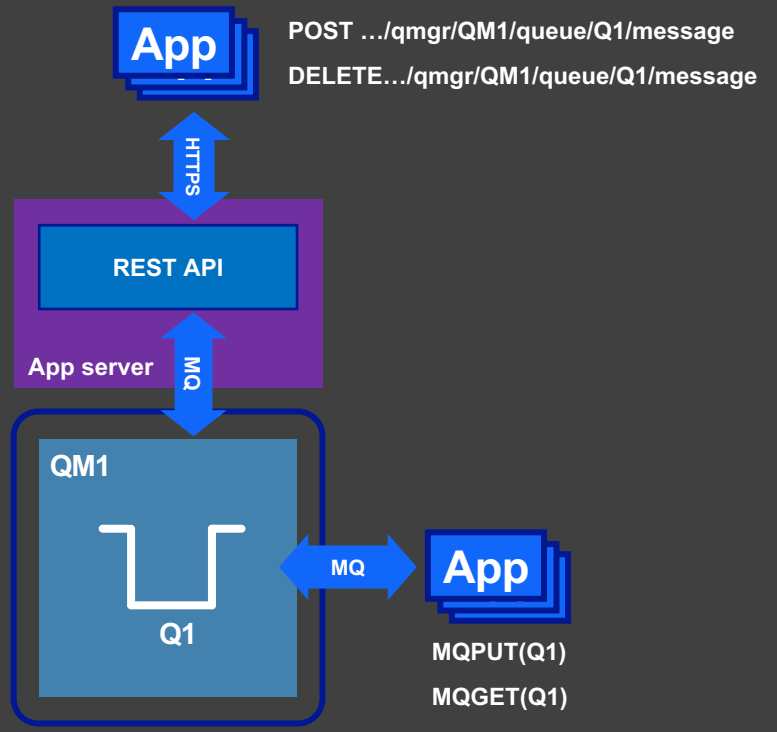
Many users just want the simplest way to get messages in and out of an MQ system.

A RESTful API gives you just that. Easily enabling messaging from just about any environment with no need for an MQ client or platform/language limitations.

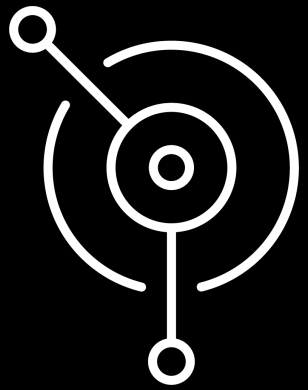
The new HTTP server support in MQ 9.0.x provides the platform for a properly integrated REST API solution.

Simple synchronous point-to-point support.

MQ support for z/OS Connect also provides a similar mechanism.



GSE UK Conference 2019



# IBM **MQ**

## Application Essentials: Anti-Patterns

# Application design considerations

Application design can affect:

- Performance
- Availability
- Reliability
- Scalability

... for the application, but potentially also a queue manager

# Anti-Pattern: Repeatedly connecting

Common development approach:

Step 1: Write code to get/send a message
Step 2: Repeatedly call code written in step 1

Common pitfalls:

- ✗ Repeatedly connect and disconnect
- ✗ Repeatedly open and close queues/topics

Impact:

- Poor performance for application
- Greater overhead for queue manager

Solution:

- Reuse session, context and object handles
- Use LLA to pre-load SCSQAUTH/SCSQANLx for local application connections on z/OS

```
for (int i = 1; i <= 5000; i++)
{
  context = cf.createContext();
  destination = context.createQueue("queue:///" + QUEUE_NAME);
  TextMessage message =
   context.createTextMessage("This is message number " + i + ".\n");
  producer = context.createProducer();
  producer.send(destination, message);
  context.close();
}
```

```
context = cf.createContext();
destination = context.createQueue("queue:///" + QUEUE_NAME);
producer = context.createProducer();

for (int i = 1; i <= 5000; i++)
{
  TextMessage message =
    context.createTextMessage("This is message number " + i + ".\n");
  producer.send(destination, message);
}

context.close();
```

# Anti-Pattern: Polling

Common development approach:

Step 1: Write code to get a message
Step 2: If no message available then call again

Common pitfalls:

✗ Rapidly and repeatedly fail to get a message because one is not available

Impact:

- High CPU usage for application
- Greater overhead for queue manager

Solution:

- Poll less often or wait longer for messages
- Use event driven messaging (async-consume)
  → Ideal when getting from multiple queues

```
while (true)
{
  // Attempt to get a message, waiting for up to 1ms
  Message receivedMessage = consumer.receive(1);

  ... process message ...
}
```

```
while (true)
{
  // Attempt to get a message, waiting indefinitely
  Message receivedMessage = consumer.receive();

  ... process message ...
}
```

OR

```
context = cf.createContext();
destination = context.createQueue("queue:///" + QUEUE_NAME);
consumer = context.createConsumer();

MessageListener ml = new MyMessageListener();
consumer.setMessageListener(ml);

-------

public class MyMessageListener implements MessageListener
{
  public void onMessage(Message message)
  {
    ... process message ...
  }
}
```

# Anti-Pattern: Optimistic programming

Common development approach:

Step 1: Write code to send/receive messages
Step 2: Declare success → move on to next task

Common pitfalls:

✗ Forget to write code to handle errors or exceptions

Potential impact:

- Messages are discarded or processed incorrectly
- Application is unreliable, performs badly, or crashes

Solution:

- Check return codes and handle exceptions

```
TextMessage message =
  context.createTextMessage("This important message MUST be sent");

producer.send(destination, message);

System.out.println("Sent message!");
```

```
try
{
  TextMessage message =
    context.createTextMessage("This important message MUST be sent");

  producer.send(destination, message);

  System.out.println("Sent message!");
}
catch (Exception ex)
{
  Throwable causeex = ex.getCause();

  if (causeex instanceof MQException)
  {
    ...
  }

  ...
}
```

# Anti-Pattern: Long-running UOWs

Causes:

- ✗ Processing large numbers of messages within a transaction
- ✗ Forgetting to commit and start a new transaction
- ✗ Incorrect sync-point option – default is different on z/OS

Potential impact:

- Increased resource requirements for the queue manager (storage, locks, recovery log)
- Can cause a queue manager outage in extreme circumstances
- Application errors if max UOW size reached

Related notes:

- ✓ Keep transactions as small as possible – do the actions really need to be performed atomically?
- ✓ Usually best to use a sync-point when processing persistent messages
- ✓ Cheaper to process multiple persistent messages in a single UOW – no need to force the log after each put or get.
- ✓ UOW size limited by queue manager MAXUMSGS parameter
- ✓ Remember to check whether transactions are committed successfully

# Anti-Pattern: Exclusivity

Applications can open a queue to get messages exclusively

- ✓ Can be ideal when sequence is important

Potential impact:

- ✗ Harder to scale application
- ✗ Harder to exploit shared queues on z/OS

Can another technique be used instead, such as partitioning using a correlation ID?

# Anti-Pattern: Hard-coded values

Avoid hard-coding queue manager, queue and topic names

Potential impact:

- ✗ Harder to reuse application
- ✗ Harder to scale application
- ✗ Harder to promote from dev → prod

# Anti-Pattern: Incorrect persistence

Persistent messages are ideal for:

- ✓ Simplifying application logic
- ✓ Auditability

However…

- Persistent messages require additional processing for recovery
- Performance can be reduced

If an application can detect and recover from lost messages in the event of a failure then persistence might not be necessary.

# Other considerations

Long-running applications might consume unnecessary system resources

→ Triggering an application when messages arrive might be more efficient

Is the message encoding important?

→ If so, good idea to request data conversion when getting messages

Be explicit about what options are required – be careful about platform-specific default values

# Notices and disclaimers


Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those
customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

# Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM’s products. IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.
The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

# Please submit your session feedback!

- Do it online at http://conferences.gse.org.uk/2019/feedback/JM

- This session is JM